Heidelberg University | Heilbronn University of Applied Science | Taipei Medical University

IMPLEMENTATION OF AN INTERACTIVE PATTERN MINING FRAMEWORK ON ELECTRONIC HEALTH RECORD DATASETS

Master Thesis

Submitted to the Department of Medical Informatics in partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

Written and presented by

Vitaliy Ostapchuk

Submitted: Monday, May 6, 2019

Referees: Prof. Dr. Daniel Pfeifer
Heilbronn University of Applied Science

Prof. Dr. Shabbir Syed Abdul
Taipei Medical University

AFFIDAVIT

I hereby confirm that my thesis entitled “Implementation of an Interactive Pattern Mining Framework on Electronic Health Record Datasets“ is the result of my own work. I did not receive any help or support from commercial consultants. All sources and/or materials applied are listed and specified in the thesis.

Furthermore, I confirm that this thesis has not yet been submitted as part of another examination process neither in identical nor in similar form.

Signature: _____

Date: Monday, May 6, 2019

Matriculation numbers:

197744 – Heilbronn University of Applied Science

3195255 – Heidelberg University

ACKNOWLEDGMENTS

I would like to extend thanks to the many people, in many countries, who so generously supported me with the work presented in this thesis.

Profound gratitude goes to my supervisor, Prof. Daniel Pfeifer of Heilbronn University. My thesis has been an amazing experience and I am sincerely thankful, not only for his academic support, but also for the encouragement and giving me the opportunity to go abroad to Taiwan.

Similar, special mentions goes to my supervisor in Taiwan, Prof. Syed Abdul Shabbir of Taipei Medical University. I am particularly indebted for his constant faith in my work, his guidance and for his support when so generously hosting me in Taiwan. I have very fond memories of my time there.

I am very grateful to the members of the Graduate Institute of Biomedical Informatics for welcoming me in the team and for their support. I would like to especially acknowledge Prof. Yu-Chuan Li (Jack), for the hospitality, advice and for sharing his expertise.

I am also hugely appreciative to Dr. Antonio Millana Martínez of Universitat Politècnica de València, for his advice and expertise in the early stages of this thesis.

My deep appreciation goes out to Friederike Sottek (B.Sc.) and Maximilian Kurscheidt (B.Sc.) for their advice and feedback on my work.

Finally, but by no means least, thanks go to all my new and old friends and family for almost unbelievable support. They are the most important people in my world and I dedicate this thesis to them.

ABSTRACT

Large collections of electronic patient records contain a broad range of clinical information highly relevant for data analysis. However, they are maintained primarily for patient administration, and automated methods are required to extract valuable knowledge for predictive, preventive, personalized and participatory medicine. Sequential pattern mining is a fundamental task in data mining which can be used to find statistically relevant, non-trivial temporal dependencies of events such as disease comorbidities. This works objective is to use this mining technique to identify disease associations based on ICD-9-CM codes data of the entire Taiwanese population obtained from Taiwan's National Health Insurance Research Database.

This thesis reports the development and implementation of the Disease Pattern Miner – a pattern mining framework in a medical domain. The framework was designed as a Web application which can be used to run several state-of-the-art sequence mining algorithms on electronic health records, collect and filter the results to reduce the number of patterns to a meaningful size, and visualize the disease associations as an interactive model in a specific population group. This may be crucial to discover new disease associations and offer novel insights to explain disease pathogenesis. A structured evaluation of the data and models are required before medical data-scientist may use this application as a tool for further research to get a better understanding of disease comorbidities.

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

4P-medicine	predictive, preventive, personalized and participatory medicine
ARM	association rule mining
EHR	Electronic Health Record
EL	Expression Language
FIM	frequent itemset mining
FP-Growth	frequent pattern growth
FP-Tree	frequent-pattern tree
ICD	International Statistical Classification of Diseases and Related Health Problems
IDE	integrated development environment
i-extension	item-extended pattern
JSP	JavaServer Pages
JSTL	JavaServer Pages Standard Tag Library
NHIRD	Taiwan's National Health Insurance Research Database
SDB	sequence database
s-extension	sequence-extended pattern
SPM	sequential pattern mining
TDB	transaction database
WHO	World Health Organization

1. INTRODUCTION

Rapidly evolving information and computer technology have transformed healthcare organizations by improving health services and knowledge management infrastructure. Large amounts of data, driven by record keeping, compliance, and regulatory requirements, and patient care is collected and accessible electronically. Data mining provides useful techniques to automatically extract valuable, non-trivial knowledge or find interesting patterns in the available datasets. Sequential pattern mining is a fundamental task in data mining to analyze sequences of events. The objective of this project is utilizing the methods of sequential pattern mining in a domain-specific interactive knowledge discovery framework.

1.1 BACKGROUND

Epidemiology is the study of the distribution and determinants of health-related states or events, health problems, and their comorbidities. Many data-driven analyses on Electronic Health Record (EHR) have shown benefits for clinical practice. In particular, this data-driven analysis can broadly be classified into two categories: discovery of diseases that often co-occur in patients, referred to as disease comorbidity or disease-disease associations, and identify disease-gene associations, which are genetically the most relevant [1, 2]. Understanding relationships between diseases, such as comorbidities or disease-disease associations, gained the attention of many researchers during the past years. Especially the predictive, preventive, personalized and participatory medicine (4P-medicine) may benefit from these knowledge discovery methods [1, 2]. Traditional statistical and network analysis methods have been augmented by data mining and machine learning techniques to obtain more complex and interesting knowledge. These insights help to understand the correlations between diseases or disease-groups, provide retrospective information on a patient trajectory, support prospective therapy decisions, and give epistemological information on conditions which are relevant in certain population groups. However, disease comorbidity patterns research is often limited by small datasets or the focus on specific diseases and patient populations. Further, the sequential order of diseases is

often neglected during the analysis. This may be crucial to discover new disease associations and offer novel insights to explain disease pathogenesis.

1.2 CONTEXT

This work is using the unique opportunity to access high-quality, dense dataset of electronic patient records containing diagnostic and medication codes of the entire Taiwanese population. The collection is obtained from Taiwan's National Health Insurance Research Database. Nevertheless, data is an asset only if processed by the right algorithms and visualized in a meaningful and interpretable way. The value comes from the possibility to extract useful information or knowledge. Data mining uses some specialized computational methods to discover meaningful and valuable structures in the data. The objective is to find potentially useful conclusions and support informed decisions. Compared to the usual statistical computations data mining defines a process with a set of tasks which help to discover novel patterns or generalizations. Pattern mining is a specialized field in data mining which focuses on determining non-trivial relationships between items and itemsets. Pattern mining algorithms improved significantly in the past two decades and showed promising possibilities for different applications. This provides the opportunity to utilize the state-of-the-art algorithms in an interactive knowledge discovery framework in the medical domain. This work focuses on sequential pattern mining to discover and model sequential relationships between diseases. Given the large collection of electronic health records the data mining algorithms are used to find interesting insights to disease associations which are common in a specific gender- or age-groups.

1.3 OBJECTIVE

The primary objective of this project shall be to explore the use of sequence mining techniques for analysis of a large set of electronic health records. This works focus is to develop and implement a framework which may help to find, model and explore sequential disease patterns. This may help to gain knowledge about disease associations, patient trajectories, and long-term treatment processes.

The objectives of this project are:

- To address the domain-specific challenges, deal with the numerous extensions for sequential pattern mining algorithms, and generate reproducible results this

work shall include a structured requirements analysis, outline the data processing pipeline and explain the framework design.

- The framework shall be developed as a Web application following a minimalistic design principle around the content and main functionality.
- The application shall support to run many of the current state-of-the-art sequence pattern mining algorithms to find frequent sequences of disease within the large dataset.
- The collection, filtering, and visualization of the found disease patterns and a model to interactively explore the disease associations shall be the main contributions to research on disease associations.
- The application design, sequential pattern mining algorithms, and the results visualizations shall be reviewed critically according to the requirements definitions.

1.4 OUTLINE

This thesis has the following structure. Chapter 2 will introduce the preliminary notions, basic concepts and related research, to give the reader a better understanding of the technical and medical background of this work. After a detailed requirements analysis in Chapter 3, the actual concept is explained in Chapter 4, followed by the realization, outlined in Chapter 5. In Chapter 6 the implemented framework and the results obtained from the mining processes are presented. Chapter 7 presents an example of a possible analysis with this framework. Further, the limitations and opportunities of the application will be discussed. Conclusion and future work will be outlined in Chapter 8.

2. BASIC CONCEPTS AND RELATED WORK

Data mining refers to the automated analysis of large datasets to discover valuable information, patterns, and rules. It has been a topic of great interest in the last decades, and therefore a large amount of research was conducted in this discipline. Data mining techniques are designed to discover hidden, unexpected and useful information which can be used to predict future outcome. Some of the most fundamental tasks are clustering, classification, and pattern mining [3]. Especially with the improvement and advancement of different algorithms, mining large datasets has become feasible yet challenging. In many domains, it is essential to consider the sequential relationship in the data to discover more important patterns.

A good example of this is health care information systems which collect large collections of electronic patient health records each time a patient visits a medical institution. This data may contain a broad range of hidden clinical information like disease associations or co-occurrences. The task of sequential pattern mining was proposed as a solution for analyzing this kind of sequential data [4, 5]. This chapter surveys previous work in sequential pattern mining as a fundamental task in data mining and disease associations analysis.

2.1 PRELIMINARIES

This section presents a formal definition of the sequence mining problem with some illustrative examples. The notation and terminology follow the standards in the related literature as introduced in the original mining task definitions of Agrawal and Srikant [5, 6], as well as the notations of Fournier-Viger et al. in the recent survey papers of itemset mining [7] and sequential pattern mining [4].

Let $I = \{i_1, i_2, \dots, i_n\}$ be a set of *n* distinct *items* and X be a non-empty *sub-set of items* such that $X \subseteq I$. Without loss of generality assume that there exists a total order on items $\prec$ such as the lexicographical order ($e.g. a \prec b \prec c$). The number of items within an itemset is called *length*. An itemset with the length $k = |X|$ items is called *k-itemset*.

A *transaction* or *event* T_t is an itemset with an additional time component t, which registers the time when the transaction was executed. A *transaction database TDB* is a set of tuples (TID, T_t), where T_t is a transaction having a *transaction-id* TID. Transactions may have the same itemset and time component, but must have a unique TID. Table I shows an example of a transaction database containing three different transactions. These transactions could, for example, represent different reported diagnoses at a particular time period. Note that some items, like b and f, seem to cooccur together in different itemsets.

Table I. A transaction database.

TID	Transaction
1	$\{a, b, f\}$
2	$\{b, f\}$
3	$\{a, b, e, f\}$

A *sequence* $\alpha = \langle T_{t_1}, T_{t_2}, \dots, T_{t_l} \rangle$ is an non-empty ordered list of transactions, where the transaction time $t_1 \leq t_2 \leq \cdots \leq t_l$. The number of items within a sequence is called the *length* of the sequence. A sequence of length $l = \sum_i |T_i|$ is referred to as *l-sequence*. An item can occur at most once in an itemset, but can occur multiple times in various transactions within a sequence.

A *sequence database SDB* is a set of tuples (SID, α), where α is a sequence associated with a *sequence-id* SID. Sequences may contain the same transactions, but must have a unique SID. Table II illustrates a sequence database with four different sequences. Note that the sequence with $SID = 2$ is composed from transactions shown in the transaction database in Table I. Each sequence could, for example, represent all reported diagnoses belonging to one patient.

Table II. A sequence database.

SID	Sequence
1	$\langle \{a, b, c\}, \{b, e\}, \{g\}, \{a, b, c, e\} \rangle$
2	$\langle \{a, b, f\}, \{b, f\}, \{a, b, e, f\} \rangle$
3	$\langle \{a, b\}, \{b\}, \{c\}, \{f, g\}, \{f\}, \{e\} \rangle$
4	$\langle \{b\}, \{c\}, \{f, g\} \rangle$

All sequences $\alpha = \langle A_1, A_2, \dots, A_n \rangle$ are called *sub-sequences* of another sequence $\beta = \langle B_1, B_2, \dots, B_m \rangle$ and β is called *super-sequence* of all α, if there exist

integers $1 \leq t_1 \leq t_2 \leq \cdots \leq t_n \leq m$ such that $n \leq m$ and $A_1 \subseteq B_{t_1}, A_2 \subseteq B_{t_2}, \ldots, A_n \subseteq B_{t_n}$. The sequence α is said to be *included* or *contained* in β. For example, the sequence $\langle\{b\},\{c\},\{f,g\}\rangle$ is included in the sequence $\langle\{a,b\},\{b\},\{c\},\{f,g\},\{f\},\{e\}\rangle$.

The number of tuples in *SDB* is called the *base size* of SDB, denoted as $|SDB|$. The *absolute support* or *frequency* of the sequence α, denoted $\sigma_\alpha = \sigma(\alpha, SDB)$, is defined as the number of super-sequences of α in the database *SDB*. The *relative support* is defined as the percentage of tuples in *SDB* that contain α (i.e., $\frac{\sigma(\alpha,SDB)}{|SDB|}$). For example, the SDB presented in Table II has a base size $|SDB| = 4$ and the sequence $\gamma = \langle\{a,b\}\rangle$ has the absolute support $\sigma_\gamma = 3$, because it appears in three different sequences ($SID = 1, 2$ & 3).

Given a user-specific positive integer σ_{min} as a threshold, termed *minimum support*, a sequence γ is a sequential *pattern* in the given SDB and is said to be *frequent* if $\sigma_\gamma \geq \sigma_{min}$. The *general sequential pattern mining problem* is to find the complete set of sequential patterns in the given SDB and satisfying the σ_{min} threshold.

2.1.1 Frequent Sets

Given the σ_{min} threshold there is always a single correct answer to the general sequential mining problem. However, discovering all sequential patterns in the naive approach by calculating the support of all possible sub-sequences and output only those meeting the σ_{min} threshold is an enumeration problem and is unrealistic to solve for most real-life sequence databases. Especially long patterns or periodic patterns may result in a not manageable search space [8, 9]. A sequence containing n items can have up to $2^n - 1$ distinct subsequences. Because of this large search space, the time complexity and potential output size of the results it is necessary to design efficient algorithms to avoid exploring all possible subsequences. Therefore, some search related definitions are presented, before introducing the specific algorithms. [7]

Closed frequent

Closed patterns are patterns which are not included in other frequent sequential patterns than itself having the same support. Discovering only closed sequential patterns instead of all frequent sequential patterns may reduce the result set significantly. However, closed sequential patterns can be used to recover all sequential

patterns and their support without accessing the original database. This property is referred to as *lossless* representation of all sequential patterns. [7]

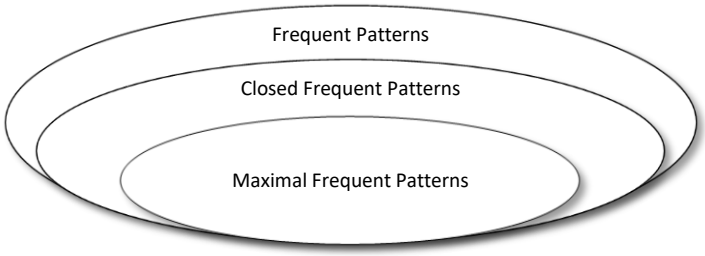


Figure 1. The relationship between frequent itemsets

Maximal frequent

A *maximal pattern* is any pattern which is not a sub-sequence of any other frequent pattern in the database [7]. As illustrated in Figure 1 closed and maximal frequent itemsets are subsets of all the frequent itemsets, but maximal frequent itemsets usually have a more compact representation. Although the number of maximal patterns can be much less than closed patterns, all frequent itemsets can be obtained without any additional scan of the original database. However, unlike the closed patterns, an additional scan is needed to recover the support values. For example, given the itemsets from the transaction database in Table I and a minimum support $\sigma_{min} = 2$, there would be seven frequent itemsets, two closed itemsets and only one maximum itemset. The results are displayed in Table III.

Table III. Frequent, Closed and Maximal Patterns from Table I.

Frequent Itemsets	**Closed Itemsets**	**Maximal Itemsets**
$\{a\} \rightarrow (\sigma = 2)$	$\{b, f\} \rightarrow (\sigma = 3)$	$\{a, b, f\} \rightarrow (\sigma = 2)$
$\{b\} \rightarrow (\sigma = 3)$	$\{a, b, f\} \rightarrow (\sigma = 2)$	
$\{f\} \rightarrow (\sigma = 3)$		
$\{a, b\} \rightarrow (\sigma = 2)$		
$\{a, f\} \rightarrow (\sigma = 2)$		
$\{b, f\} \rightarrow (\sigma = 3)$		
$\{a, b, f\} \rightarrow (\sigma = 2)$		

Generator patterns

G*enerator sequential patterns* are the set of patterns which have no sub-sequences having the same support. The set of generator patterns is a subset of all frequent patterns. Generator patterns are used to describe or characterize a group of sequences in the sequence database. They can also be combined with information from closed or maximal frequent pattern sets to provide higher accuracy in classification or to generate rules for further analysis. [7]

2.2 DATA MINING BACKGROUND

Most of the current *sequential pattern mining* (SPM) [5] algorithms have emerged from the traditional *frequent itemset mining* (FIM) or *association rule mining* (ARM) [6] and have been improved and extended by key features over time. This section presents the main characteristics and how this field of research developed since the early 1990s when Agrawal and Srikant proposed the Apriori algorithm. [7]

2.2.1 Frequent Sequence Patterns

Most real-world problems and data have a time dependency or a sequential order of events. Agrawal and Srikant address this problem by extending the original Apriori algorithm for mining subsequences in a set of sequences [5]. By using discretization techniques, it can also be applied to any time-series data. Traditional domains for this mining task are for example stock market predictions, sensor data readings, text and language analysis, customer behavior prediction and bioinformatics [4]. Before reviewing the differences between the various utilization strategies and variations of the algorithms it is important to give an overview related to all sequential pattern mining.

The assumed general total order $\prec$ represents the order of processing items in the database. It is used to follow a specific order to explore potential frequent patterns. Note that any order $\prec$ has no influence on the final result produced by the algorithm, but reconsidering the same patterns more than once is avoided.

There are two basic operations called *s-extension (sequence-extended pattern)* and *i-extension (item-extended pattern)*, which are performed by all sequential pattern mining algorithms. Both operations are used to *extend* a k-sequence to a (k+1)-sequence. These operations are formally defined as followed. Let $\alpha = \langle A_1, A_2, \dots, A_n \rangle$ and $\beta = \langle B_1, B_2, \dots, B_m \rangle$ be two sequences. α is called a *prefix* of sequence β if $A_1 = B_1, A_2 = B_2, \dots, A_{n-1} = B_{n-1}$ and A_n is equal to the first $|A_n|$ items of B_n according to the total order $\prec$. A sequence γ is said to be an *s-extension* of α with an item *x*, if $\gamma = \langle A_1, A_2, \dots, A_n, \{x\} \rangle$, i.e., α is a prefix of γ and the item *x* appears in an itemset occurring after all itemsets of α. A sequence γ is said to be an *i-extension* of β with an item *x*, if $\gamma = \langle B_1, B_2, \dots, B_m \cup \{x\} \rangle$ and the item *x* is the last item occurring in S_γ according to the total order $\prec$. For example, the sequence $\alpha = \langle \{a, d\}, \{b\}, \{c\} \rangle$ would

be a s-extension and $\beta = \langle\{a, d\}, \{b, c\}\rangle$ would be an i-extension of the sequence $\gamma = \langle\{a, d\}, \{b\}\rangle$ as the prefix and the item c as an extension.

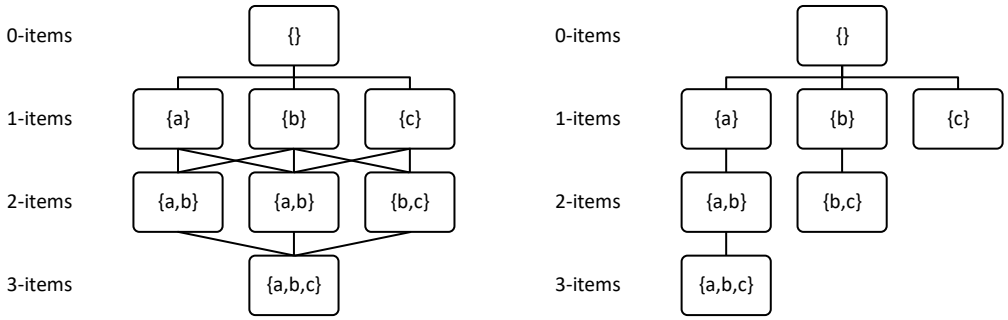


Figure 2. Breadth-first (left) vs. depth-first search (right).

The mining algorithms can be categorized in being either *depth-first* search or *breadth-first* search algorithms [4]. An example visualization of both search methods is given in Figure 2. The breadth-first search, also called *level-vise approach*, will scan the database for frequent 1-sequences, then generate all 2-sequences by performing s-extensions and i-extensions, then all 3-sequences using the 2-sequences and so on until no sequences can be generated. The enumeration problem for long sequences can have a huge search space, and the search might fail due to memory and runtime errors. Therefore most of the recent algorithms tend to explore the search space in the depth-first order. Depth-first algorithms also start with sequences containing single items, but then recursively perform i-extensions and s-extensions with only one of the sequences to generate larger sequences in each recursive step. If no more extension can be performed the algorithm backtrack, to generate other sequences with the same prefix but another item as an extension.

To reduce the computation cost of the mining task, designing an efficient algorithm requires to integrate techniques, referred to as *pruning*, to avoid exploring the whole search space. One of the most common pruning techniques is based on the so-called *apriori property*, also known as *downward-closure property* or the *anti-monotonicity property*. This property states that any non-empty subset of a frequent itemset must also be frequent. It can be also be applied to sequential patterns indicating that if the sequence α is included in the sequence β, then $\sigma_\alpha \geq \sigma_\beta$. The application of this property can greatly reduce the search space since any pattern candidate which is infrequent cannot generate frequent extensions. [6]

2.2.2 Traditional Itemset Mining Task

One of the most common tasks in data mining is to find frequent itemsets or groups of items frequently appearing together in a database. Since this knowledge has numerous real-life applications and domains like market basket analysis, bioinformatics, text mining, product recommendation, e-learning, and Web analysis, this field of data mining became a popular and active research topic. Two major approaches have been introduced to address this problem.

Apriori-Based

The algorithm proposed by Agrawal and Srikant was the first of its kind in the field of FIM. It was initially intended for analyzing customer data but is now viewed as a general data mining task. [6]

The pseudocode of the Apriori algorithm is given in Figure 3. It starts with an initial scan of a transaction database to calculate the support for each distinct item (line 1). This information is used to identify a set of all frequent items, denoted as F_1, meeting the given minimal support threshold (line 2). Then a breadth-first search is performed to find extended itemsets (lines 4-10). During the search, the Apriori first generates all potential itemsets of length k, denoted as C_k, by combining the already observed frequent itemset F_{k-1} (line 5). All candidate items which contain some $(k-1)$-itemset that is not in F_{k-1} violate the downward-closure property. They are therefore removed from C_k (line 6). The support values for all remaining itemsets is calculated by scanning the database (line 7). The search is repeated for all candidates F_k meeting the σ_{min} threshold until no new candidates can be generated.

Many other algorithms have been inspired by the Apriori algorithm and use the key feature of pruning by the downward-closure property. However, there are some significant limitations to the original algorithm. First, the Apriori generates candidates without looking at the original database, which can result in creating candidates which are not observed at all. Second, the algorithm needs to scan the database each time the candidates are evaluated. Another important limitation is the breadth-first search, which in the worst case requires to keep all patterns in memory at any time. [7]

Input: ***TDB***: Transaction database, $\boldsymbol{\sigma_{min}}$: Minimum support threshold.

Output: Set of frequent itemsets.

1. Scan the database TDB to calculate the support for all items in I;
2. $F_1 = \{i | i \in I \wedge \sigma_i \geq \sigma_{min}\}$; // F_1 : frequent 1-itemsets
3. $k = 2$;
4. **While** $F_k \neq \emptyset$ **do**
5. $C_k = CandidateGeneration(F_{k-1})$; // C_k : candidate k-itemsets
6. Remove each candidate $X \in C_k$ that contains $(k-1)$-itemset that is not in F_{k-1};
7. Scan the database TDB to calculate the support of each candidate $X \in C_k$;
8. $F_k = \{X | X \in C_k \wedge \sigma_X \geq \sigma_{min}\}$; // F_k : frequent k-itemsets
9. $k = k + 1$;
10. **End**
11. **Return** $\cup_{n=1 \ldots k} F_n$

Figure 3. The Apriori algorithm. [6, 7]

FP-Growth

Algorithms like Eclat [10] improved upon the Apriori by using a depth-first search. However, many of the bottleneck features, especially the candidate generation, of Apriori remained unsolved until the *frequent pattern growth* (FP-Growth) algorithm was introduced by Han, Pei, and Yin in 2000 [11].

To avoid generating candidates which do not appear in the database the FP-Growth algorithm is using an extended prefix-tree structure, called *frequent-pattern tree* (FP-Tree). The mining task is transformed from mining the database into mining this compact tree. The tree structure provides a compact view of the data with an additional header table for every distinct item that satisfies the support threshold. Additionally, the concept of a *projected database* is introduced to reduce the explored search space effectively. The following section outlines these concepts of FP-Growth algorithms. [7, 12]

The core method uses a recursive divide-and-conquer strategy to retain associations between items without candidate generation. The pseudocode of the FP-Growth algorithm is given in Figure 4. The initial input is the transaction database TDB, a minimal support threshold σ_{min} and an empty itemset X. It starts with an initial scan of the TDB to find the set I of frequent items given the σ_{min} threshold (line 2). For each found item i, the itemset $X \cup \{i\}$ has to be a frequent itemset and therefore is added to the output (line 4). Since not all items of the TDB can be appended to generate an extended frequent itemset, a projected database (concept outlined in the example

below) on the itemset $X \cup \{i\}$ is created to find only frequent itemsets which appear in the TDB (line 5). This projection is then used to perform a recursive depth-first search on the itemset $X \cup \{i\}$ with respect to the minimum support threshold σ_{min} (line 6). After the recursive search, all items in *X* are frequent itemsets. Note that the database scan is reduced to a minimum cost, since the information needed for extending the itemsets is obtained from the very compact projected database. Further, the generated candidate itemsets can only be sets which are truly observed in the database, thus avoids considering many itemsets which do not appear at all. [7]

Input: ***TDB***: Transaction database, $\boldsymbol{\sigma_{min}}$: Minimum support threshold, ***X***: the current itemset (initially $X = \emptyset$).

Output: Set of frequent itemsets.

1. Function $PatternGrowth(TDB, \sigma_{\min}, X)$
2. Scan the database TDB to find the set I of all frequent items in TDB;
3. **Foreach** item $i \in I$ **do**
4. Output $X \cup \{i\}$; // X ∪ {i} is a frequent itemset
5. $TDB' = Projection(TDB, i)$; // Create a projected database of X ∪ {i}
6. $PatternGrowth(TDB', \sigma_{\min}, X \cup \{i\})$; // Recursive call to extend X ∪ {i}
7. **End**

Figure 4. The FP-Growth algorithm. [7, 11]

To illustrate these steps the example TDB from Table I is used once more with the minimum support threshold $\sigma_{min} = 2$. After an initial scan of the database, the frequent 1-itemsets are $\{a\}$ with $\sigma = 2$, $\{b\}$ with $\sigma = 3$ and $\{f\}$ also with $\sigma = 3$. The algorithm will create a projected database for each of these itemsets to perform the mining task. The projected database TDB' of any item *i* is defined by an transaction set where *i* appears, but all preceding items according to the general total order $\prec$ and the item *i* itself have been removed. To find the frequent extensions the algorithm will scan the projected database for each itemset and estimate the support values. The algorithm will follow the depth-first search recursively for each frequent set found in the projection until all frequent sets have been explored. An example for the search in the projected databases of $\{a\}$ is shown in Table IV and Table V.

Table IV. Projected database of $\{a\}$.

TID	Transaction	σ
1	$\{b, f\}$	2
3	$\{b, e, f\}$	1

Table V. Projected database of $\{a, b\}$.

TID	Transaction	σ
1	$\{f\}$	2
3	$\{e, f\}$	1

Pattern-growth algorithms offer many improvements over the original Apriori design. However, there are also limitations and optimizations which have been addressed in related research. Especially the cost to create a projected database was evaluated and optimized by a *pseudo-projection*, which consists of pointers on the original database. Note that many other improvements, e.g., Map-Reduce paradigm, can be integrated into a pattern-growth algorithm. [7]

2.2.3 Efficiency Improvements by Novel Designs

Itemset mining and related topics have been an active research topic for almost thirty years and still offer countless opportunities for research. Especially a lot of effort was made to enhance the performance by novel data structures and algorithm designs. This is one of the most important problems since the mining task can be very time-consuming. Much of the past research has focused on reducing the search space (e.g., early pruning techniques), operations on the data (e.g., IDLists) or a smart way of representing the itemsets to save memory (e.g., bitvector representation). Some of the algorithms introduced in the last decade propose designs which allow running parallel mining task on distributed systems, GPU's or multicore to increase speed and scalability. Furthermore, there have been plenty of extensions and variations from the original field of FIM, for example, high-utility itemset mining, rare-pattern mining or association rule mining. [4, 13–15]

2.2.4 Extensions

Before introducing specific algorithms some necessary extensions and techniques are described, which may be included in the mining task. Many of these extensions are used as some constraint to limit the search space and improve mining efficiency [8, 16]. This section will only outline extensions which are used in the application developed as part of this work.

Length Constraint

One of the easiest ways to limit the search space is by specifying the minimum and maximum length of the candidate patterns. This constraint may be useful to find more meaningful sequences without getting lost in the depth of the search tree. [16]

Item Constraint and Regular Expression Mining

Specifying items which have to or should not be present in the candidate itemsets is a powerful way to limit the results to a user-specific focus. This constraint can be extended to perform mining based on regular expressions over the set of items. [17]

Time, Time-Window or Duration Constraint

This constraint is defined only in sequential pattern mining or time series mining since it requires some time-stamp in the database. The candidate patterns are pruned such that the time difference between the first and last transactions must not be longer or shorter than a given period. [8, 18]

Gap Constraint

This constraint is also defined only for sequential pattern mining tasks. It requires that the time difference between any transactions in the candidate pattern is not longer or shorter than the given gap interval. This constraint type is prevalent in text and language analysis since words in a sentence often have direct relationships to their immediate neighbors. [16]

Weighted Pattern Mining

This approach extends the traditional frequent pattern mining problem by giving each item a weight instead of treating each item uniformly. Therefore each item in the database has different importance for the outcome. For example, profits of retail items used in market basket analysis or the term frequency-inverse document frequency are common weights to find more interesting frequent itemsets. [19]

Multi-dimensional Mining

Real world application datasets may have associations from multidimensional space. For example, customer purchase behavior often depends on local or regional events, time, date, customer group, and others. Finding patterns associated with multi-dimensional information is a solvable problem. But, compared to the traditional FIM task, the proposed methods suffer in terms of performance and efficiency. [20, 21]

High-Utility Mining

Similarly to the weighted pattern mining the problem of high-utility itemset mining, also referred to as simple utility mining, extends the traditional FIM problem by item weight. Additionally, utility mining also takes into account the number of items in each transaction. Especially for customer basket analysis, this kind of mining

became very popular in the last decade, since discovering frequent high-utility itemsets offers a smart way to analyze the cooccurrences of items and maximize the profit of each transaction. [22, 23]

Uncertain Pattern Mining

Various real-life applications have probabilistic datasets of uncertain data. Uncertain frequent pattern mining extends the traditional FIM to address this kind of problem. Some of the recent research in this field suggests combining the traditional mining task results with some kinds of density-based clustering methods to deal with uncertain data and outliers. [24]

2.3 Extensions for Big Data Mining

Different kind of efficient pattern mining algorithms have been implemented, yet not all of them are scalable enough to deal with the "*Big Data"* datasets collected nowadays. There is an obvious need to develop tools which can complete the mining task concerning the time and memory complexity of the problem [25]. This section outlines the main concepts used by some of the algorithms included in the framework.

2.3.1 Parallelism

Some of the serial tasks, e.g., the computation of support of generated candidates, can be parallelized. However, this naive approach relies on assumptions like an unlimited process memory and concurrent data operations. Memory scalability, task partitioning, and load balancing are the most critical problems to solve while designing a parallel algorithm. [25, 26]

Memory scalability is the problem of fitting the data in memory. Note that, even if the input data is small enough to be loaded into memory, the process itself might fail, e.g., the candidate generation or the data structures used by the algorithm may be too large. [25]

To design efficient parallel algorithms the process has to be divided into smaller units of work, called *tasks*, where each task is independent and can be executed concurrently. To archive an efficient design the computational work by each task has to be distributed equally while minimizing the inter-processor communication cost between the tasks. The mining tasks tend to be highly irregular depending on the input dataset and the algorithm parameters chosen by the user. Therefore, the computational

work assigned to a task can lead to imbalanced processing. Dynamic load balance methods attempt to actively assign work to a process and minimize the idle time. [25]

2.3.2 Distributed Systems

While parallelization attempts to execute processes concurrently on the same machine, a distributed system consists of multiple machines working together but appear as a single computer to the end user. The machines share information and state through messages, operate concurrently and do not depend on each other. Two common paradigms, *Message Passing* and *Map-Reduce*, are used to develop memory distributed systems. Especially the more recent Map-Reduce paradigm is specifically designed for working on Big Data sets and was used in some of the recent mining algorithms. [27, 28]

2.4 SEQUENTIAL PATTERN MINING ALGORITHMS

As far as the literature review indicates, the SPMF data mining library [29] is the largest collection of open-source implementations of sequential mining algorithms. It provides more than 120 different pattern mining algorithms implemented in Java and can be integrated into other applications or run as standalone software [14]. Most of the algorithms used in this work are available through the SPMF library.

Many of the algorithms developed in the field of sequential pattern mining are based on previous work and improved upon limitations of other algorithms. Since there is a large number of different extensions and parameters which are addressed in specific mining tasks this section will only discuss the main deviations and differences from other implementations. An overview of all sequential pattern mining algorithms used for the implementation in this project is shown in Table VI and is surveyed in the following section.

Agrawal and Srikant formally defined the sequential pattern mining problem in transactional databases in 1995 as an extension of their original Apriori algorithm. Progress in barcode technology has made it possible to collect and store massive amounts of market basket data, which was the primary purpose of this analysis. They introduced the basic concept of association rules, which refer to intra-transaction patterns, and sequential patterns, which relate to inter-transaction patterns. The original Apriori algorithm was already capable of discovering association rules within

itemsets. To find patterns in a sequence of itemsets they introduced and evaluated three new algorithms – AprioriAll, AprioriSome, and DynamicSome. [5]

Table VI. Sequence mining algorithms overview.

Year	Algorithm	Constraint	Features
1995	AprioriAll [5]		First algorithm for sequential pattern mining. Adoption of original Apriori algorithm.
2002	SPAM [30]		Depth-first sequence mining on vertical bitmap representation of data.
2003	CloSpan [31]	Closed	FP-Growth algorithm for discovering closed sequential patterns.
2003	TSP [32]	Closed	Mining top-k frequent sequential patterns.
2004	BIDE [33]	Closed	Mining closed sequential patterns without candidate maintenance.
2013	ClaSP [34]	Closed	Mining closed patterns on a vertical data representation with additional pruning.
2013	MaxSP [35]	Maximal	Mining maximal sequential patterns.
2013	MG-FSM [36]		Sequence mining algorithm with the gap-threshold build on Map-Reduce paradigm.
2013	TKS [37]		Mining top-k frequent sequential patterns on a vertical data representation.
2014	CM-ClaSP [38]	Closed	Co-occurrence pruning as modification to original ClaSP.
2014	CM-SPAM [38]		Co-occurrence pruning as modification to original SPAM
2014	VMSP [39]	Maximal	Mining maximal sequential patterns on a vertical data representation.
2015	CloFAST [40]	Closed	Closed sequence mining algorithm based on sparse id-lists pruning.
2016	LASH [41]		Sequence mining with hierarchies built on MapReduce paradigm.
2016	Skopus [42]		Mining exact top-k sequential patterns under leverage.

The SPAM algorithm proposed by Ayres et al. in 2002 was one of the first algorithms which utilized a depth-first search strategy for mining sequential patterns. Additionally, the algorithm uses a vertical bitmap representation of the data for effective counting of the support values and candidate generation. [30]

CloSpan is a pattern-growth algorithm for discovering closed sequential patterns in sequence databases, proposed by Yan et al. in 2003. The main concept used by the algorithm is the equivalence of projected databases. This concept was shown to be effective as a method to effectively prune the search space. [31]

Sequential pattern mining often generates an exponential number of patterns if the database consists only of long frequent sequences. Also, the outcome of the mining task heavily depends on the σ_{min} value – small values may lead to many patterns which are not relevant, whereas big values may not show any outcome at all. Tzvetkov et al. try to address these limitations by the algorithm called TSP. It takes two additional parameters, minimum pattern length (min-l) and the desired number (top-k) of sequential patterns with the highest overall support, to reduce the results to a relevant but manageable size. [32]

The sequential pattern mining algorithm BIDE was proposed by Wang et al. in 2004. BIDE introduces a concept for mining closed sequences without candidate maintenance and was shown to outperform previous work in terms of memory consumption significantly. [33]

In 2013 Gomariz et al. proposed the sequential pattern mining algorithm called ClaSP. It combines the vertical database layout with several efficient search space pruning methods to mine closed patterns. [34]

Fournier-Viger et al. proposed MaxSP in 2013 – a maximal sequential pattern mining algorithm without candidate maintenance. Mining frequent patterns degrade the performance of the mining task in terms of execution time and memory requirement and become worse with longer sequences. Mining only the maximal patterns may help to reduce the number of patterns to the more meaningful ones. However, previous algorithms needed to maintain a large number of intermediate candidates in main memory for this mining task. MaxSP neither produced redundant candidates nor stores intermediate candidates in main memory and was therefore shown to be efficient. [35]

MG-FSM was proposed by Miliaraki et al. in 2013 as a highly scalable frequent sequence mining algorithm build on the Map-Reduce paradigm. Since the primary purpose of this algorithm is *n-gram* mining in large-scale text datasets, it is the first distributed FSM algorithm that supports a general gap constraint. [36]

Similar to the TSP the TKS algorithm proposed by Fournier-Viger in 2013 aims to discover only the top-*k* sequential patterns, where *k* is the number of sequential patterns to be found and is set by the user. TKS uses a vertical database representation, a basic candidate generation procedure as SPAM and several effective strategies to prune the search space. TKS outperforms TSP in terms of execution time and memory usage, especially on dense datasets. [37]

Algorithms which use a vertical database format (e.g., ClaSP or SPAM) provide advantages in candidate generation and support count without rescanning the database. However, an important performance bottleneck remained since they use a generate-candidate-and-test approach. In 2014 Fournier-Viger et al. propose a new effective candidate pruning mechanism for vertical sequential pattern mining algorithms. It is based on item co-occurrence information, which is stored in a new data structure

named *Co-occurrence MAP*. The resulting algorithms are named with "CM-" as a prefix, e.g., CM-ClaSP, CM-SPAM. [38]

The VMSP algorithm proposed by Fournier-Viger et al. in 2014 is the first algorithm for mining maximal sequential patterns on vertical data representation. To prune the search space and identify the maximal patterns efficiently the algorithm uses three different strategies – EFN (Efficient Filtering of Non-maximal patterns), FME (Forward-Maximal Extension checking) and CPC (Candidate Pruning by Co-occurrence map). [39]

Fumarola et al. propose an algorithm, named CloFAST, for mining closed sequential patterns in 2015, which uses a novel representation of the dataset based on sparse id-lists and vertical id-lists. These data structure properties allow both to check the closure property and prune the search space in a single step. CloFAST is empirically proven to outperform the state-of-the-art algorithms, especially when mining long closed sequence patterns. [40]

In 2016 Beedkar and Gemulla introduce a new algorithm named LASH, for mining sequential patterns in the presence of hierarchies. Additionally to the sequence dataset LASH allows to input information about the hierarchy of items, e.g., "Barack Obama" may generalize to "politician" to "person". This generalization can be used to find sequential patterns and relations on multiple levels, which otherwise would be hidden. LASH is closely related to the MG-FSM algorithm. It is also built primarily for text mining on the Map-Reduce paradigm and supports the gap constraint. [41]

Petitjean et al. introduce leverage as a new measure of interest and propose a new sequence mining algorithm, named Skopus, to discover the top-k patterns under this measure. Most frequent patterns in real-world datasets are not interesting since they simply correspond to the permutation of the most frequent items. For example, given a large text as the dataset, the results of a sequence mining task would show that the most frequent patterns are permutations of stop-words like "and", "to" and "of". This illustrates that the support is not a good measure for interestingness of the patterns. The leverage measure shows if the pattern is more frequent than can be explained by an independence assumption of the sub-patterns it is composed of. It is shown that leverage may be used to discover more relevant patterns, consistent with the definition of expected support. [42]

2.5 DATA MINING IN HEALTHCARE

All public and private health institutes are producing and collecting enormous amounts of data. There is an obvious need to provide automated tools for discovery and analysis of useful information in an interpretable way. Several studies propose novel methods which support administrative and medical decision estimation, e.g., decisions regarding treatments, disease prediction, health policies. Most of the common data mining techniques such as classification, clustering, and regressions are crucial for making decisions regarding patients' health. Especially in combination with the fast-growing trend of machine learning diagnostic methods and treatments are improving rapidly. [43, 44]

2.5.1 ICD-9-CM Codes

Some of the challenges are important to point out when working with healthcare datasets. One of the most significant is to acquire the precise and complete set of data. Due to its nature, the medical data is complex, heterogeneous and collected from different sources while quality, accuracy and privacy concerns have to be maintained during the whole process. In recent years standardized datasets became available and publicly accessible. [44]

The International Statistical Classification of Diseases and Related Health Problems, commonly referred to as ICD, was established by the World Health Organization (WHO) to track morbidity and mortality rates across the world. It provides a unique, up to five characters long, alpha-numeric code to classify diseases and a wide variety of signs, symptoms, abnormal findings, diseases and others. The most recent version ICD-11 (11^{th} Revision) was released in June 2018. [45]

The ICD-9-CM (9^{th} Revision, Clinical Modification) is the U.S. health system's adaptation which was created primarily for billing purposes. In clinical practice, ICD-9-CM codes have been widely used as a record of patient diagnoses, symptoms, and conditions. [46]

The ICD-9-CM version consists of 22,401 distinct codes arranged hierarchically in 19 chapters, also referred to as groups, e.g., *"Infectious And Parasitic Diseases"* or *"Diseases Of The Digestive System"*. The first digit of the code is alpha-numeric, followed by a minimum of two and a maximum of four digits. The first three characters define the category of the disease, disorder, infection or symptom. For example, V83

encodes *"Genetic carrier status"*, 250 encodes *"Diabetes mellitus"*, 250.1 encodes *"Diabetes with ketoacidosis"*, and 250.13 encodes *"Diabetes with ketoacidosis, type I [juvenile type], uncontrolled"*.

Table VII. Statistics for the full dataset.

AGE	GENDER	TRANSACTIONS	FILE SIZE	DISTINCT PATIENT IDS	DISTINCT ICD-CODES	VALID SEQUENCES	MAX SEQ-LENGTH	AVERAGE SEQ-LENGTH
0	MALE	86.309.991	2.502 GB	1.827.447	1.406	1.772.877	204	40,89
	FEMALE	72.492.529	2.095 GB	1.677.365	1.349	1.620.468	167	39,35
1	MALE	33.127.373	966 MB	1.678.415	1.334	1.590.044	189	25,36
	FEMALE	34.277.943	999 MB	1.595.057	1.345	1.528.023	200	28,36
2	MALE	28.244.237	834 MB	1.767.163	1.346	1.598.789	198	21,62
	FEMALE	52.512.226	1.543 GB	1.780.095	1.450	1.721.742	213	34,67
3	MALE	37.314.053	1.111 GB	1.737.715	1.376	1.593.313	216	26,90
	FEMALE	58.125.055	1.717 GB	1.765.866	1.494	1.698.355	224	36,80
4	MALE	42.685.075	1.289 GB	1.577.320	1.392	1.457.387	216	33,37
	FEMALE	60.506.355	1.818 GB	1.631.968	1.521	1.555.060	236	42,75
5	MALE	32.868.519	1.010 GB	898.150	1.378	844.497	271	43,43
	FEMALE	46.933.328	1.444 GB	930.496	1.446	893.463	253	57,70
6	MALE	38.143.378	1.189 GB	692.061	1.414	666.161	250	57,27
	FEMALE	46.048.707	1.441 GB	711.096	1.437	689.877	258	67,40
7	MALE	38.797.169	1.226 GB	532.308	1.407	517.457	275	67,23
	FEMALE	32.468.511	1.028 GB	427.821	1.403	416.188	261	72,09
8	MALE	8.743.841	277 MB	133.480	1.249	128.422	270	60,43
	FEMALE	8.928.949	283 MB	141.225	1.250	135.707	287	61,13
9	MALE	170.066	5 MB	4.769	758	4.313	186	39,81
	FEMALE	274.376	9 MB	8.515	813	7.565	287	37,64
0-9	MALE	346.403.702	10.409 GB	10.848.828	1.644	10.173.260	227,50	41,63
0-9	FEMALE	412.567.979	12.376 GB	10.669.504	1.718	10.266.448	238,60	47,79
0-9	ALL	758.971.681	22.785 GB	21.518.332	2.179	20.439.708	233,05	44,71

2.6 DATASET

The dataset was obtained from Taiwan's National Health Insurance Research Database (NHIRD). A mapping tool was used to convert the claimed records to a transaction dataset with ICD-9-CM disease codes. It contains more than 758 outpatient

visits and 2179 distinct disease categories, that were observed in the entire Taiwanese population of over 21 million individuals. The subjects were followed for 36 months from January 1st, 2000 through December 31st, 2002. Each was attributed to a specific gender- and age-group, e.g., male and female, aged 0 to 9, 10 to 19, … , and 90 to 99 years. Detailed statistics of the dataset are shown in Table VII.

2.7 WEB APPLICATION TECHNOLOGIES

This section introduces the concepts of Web technologies used for this work. Nowadays, there exist various ways to develop a Web application. One of the most common to create dynamic Web applications in Java is provided by the Java™ EE Specification API's [47]. An overview of the Servlet & JavaServer Pages (JSP) technologies is shown in Figure 5 and outlined in the following section.

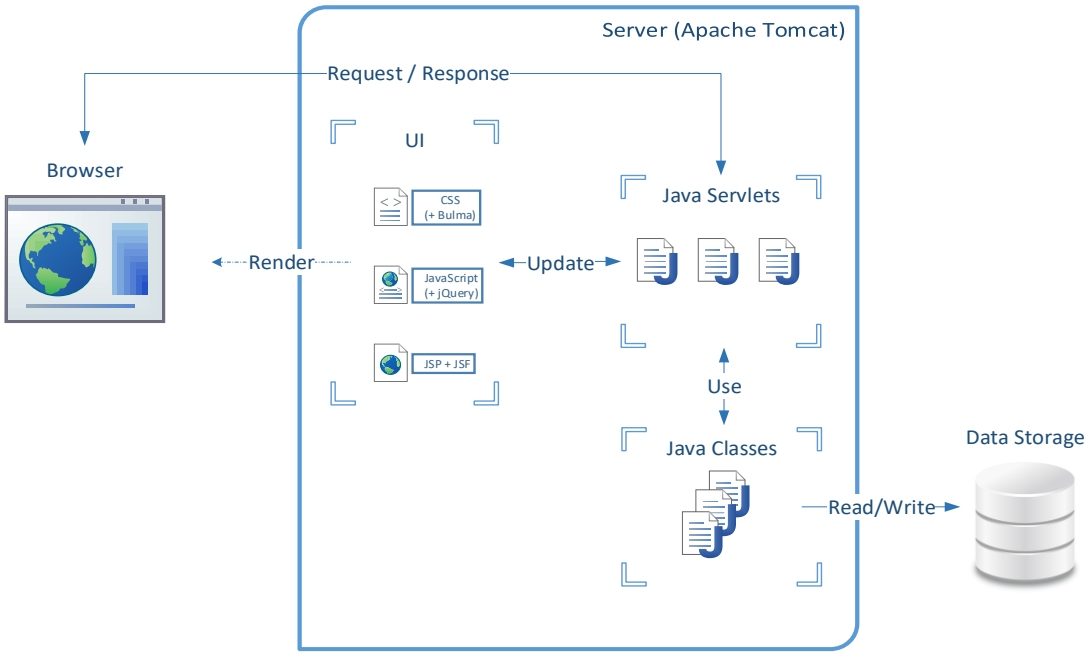


Figure 5. Overview of the basic Servlet & JavaServer Pages Web application.

The Apache Tomcat server is used as a Web servlet container on which the application is deployed. When a Web application is loaded, the `ServletContainer` creates the `ServletContext` once and keeps it in the memory. A client can access the application through a Web browser. The server listens to these requests on a defined HTTP port. The `ServletContainer` then creates new `HttpServletRequest` and `HttpServletResponse` objects and passes them down to the `Servlets`. The `Servlets` represent the controller component, which is used to invoke actions by the client, access the underlying model and pass information to the view. Standard Java classes

are used to implement the underlying logic, make data operations and represents the state of the application. A `Servlet` performs actions based on the `HttpServletRequest` and passes requested models to the JavaServer Pages of the Web application. JSP is a Java-based technology, which allows writing templates in client sided languages, e.g., HTML, CSS or JavaScript. Additionally, JSP supports tag-libraries, e.g., JavaServer Pages Standard Tag Library (JSTL), and Expression Language (EL). This makes it a powerful technology to dynamically control the routing, output and enables resolution of Java objects and methods. To implement a modern, responsible and modular interface the open source CSS framework Bulma is used [48]. The open source JavaScript library jQuery is used for some client sided HTML manipulations, event handling, CSS animation and Ajax [49].

3. REQUIREMENTS ANALYSIS

This chapter discusses the requirements for the development of the Disease Pattern Miner – a domain-specific interactive knowledge discovery framework. Identifying the project's objectives and requirements is an important step in the process of developing a new (Web-based) application is. It is essential to include and adapt the requirements to both experts domain knowledge and knowledge derived during the mining and modeling process.

3.1 REQUIREMENTS

This section briefly lists the main functional and non-functional requirements derived from the supported scenarios as well as the additional requirements and specifications determined during the literature review and implementation. The requirements are explained in detail in Sections 3.2 and 3.3.

3.1.1 Functional Requirements

[R1] The user wants to upload a transaction dataset meeting a specified predefined format.

[R2] The application should filter the dataset.

[R3] The application should split the dataset into smaller gender-age-group transaction files.

[R4] The user wants to select the transaction files which will be used in the mining task.

[R5] The user wants to create a mining algorithm task.

[R6] The user wants to specify or change the mining task parameters.

[R7] The user wants to run the mining processes.

[R8] The application should convert the transaction files to sequence files.

[R9] The user wants to view the collected results in a single table.

[R10] The user wants to filter the results table.

[R11] The user wants to export the results table to a downloadable file.

[R12] The application should find all patients having a specific frequent pattern.

[R13] The user wants to view the most frequent codes for a specific table cell.

[R14] The user wants to view the visualization of patient trajectories for a specific frequent pattern.

[R15] The user wants to filter the trajectory model.

[R16] The user wants to export the visualized model to a downloadable file.

3.1.2 Non-Functional Requirements

[R17] The framework shall be a server sided Web application.

[R18] The system shall load all files on restart automatically.

[R19] The application should work on different browsers.

[R20] The results table may show which algorithm produced the entry.

[R21] The algorithms shall have a default set of parameters.

[R22] The algorithms run may not block any actions by the user.

[R23] The application gives feedback about the status of the mining processes.

[R24] The user interface is split in different views which can be accessed by a standard menu.

[R25] All algorithms have to be available open-source.

[R26] All algorithms solve a sequential mining task.

[R27] No multi-language support.

3.1.3 Elicitation of Additional Requirements and Specifications

[R28] Hard drive space is not restricted.

[R29] The system may process large data files or at least reduce the data to a manageable size.

[R30] The Heap and RAM shall be limited not to exceed the server capabilities.

[R31] All algorithms have to be implemented in Java.

[R32] Algorithms may take into account hierarchical constraints.

[R33] No algorithms for …

a. Itemset Mining, since time order is supposed to be taken into account.

b. Association Rule Mining, since the output is supposed to be a sequence.

c. High-Utility Mining, since no utilities are given.

3.2 MINING PIPELINE

The initial goals of this work were to apply data mining methodology to find gender- or age-group-specific patterns of disease-disease associations, which are the most relevant. Building a data processing pipeline is crucial to developing a data mining application. This allows to break down and define the process, identify the activities and describe the specific problems involved in this work. The pipeline developed as part of the initial requirements gathering is presented in Figure 6 and is explained in detail in the following section. The derived requirements are referenced within the text by number and summarized in Section 3.1.

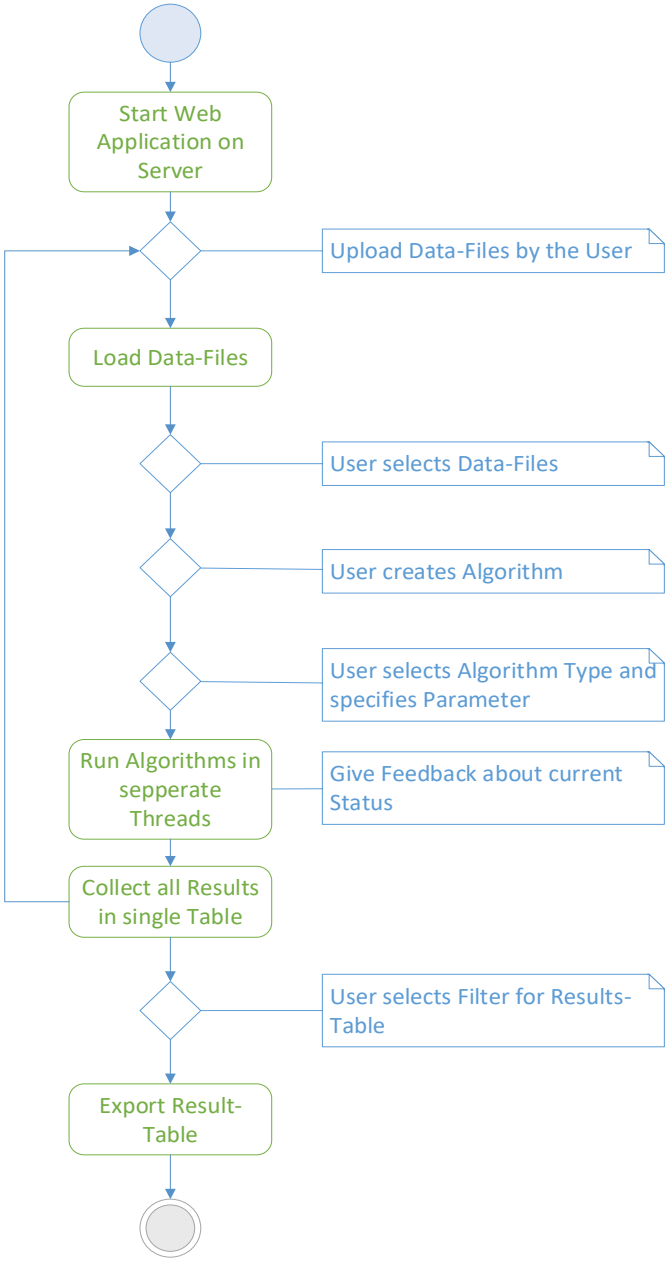


Figure 6. The data mining pipeline.

The first step in the pipeline is to set up the dependencies necessary to compile and deploy the project on a server sided Web application [R17]. After starting the Web server, the user should be able to access the application through any browser [R19][R24]. This is necessary due to two reasons. First, the user profile, who might be using the application, is limited to experts in the field of disease associations analysis and data mining. This user group requires to be able to share their data sources as well as the results. Second, the mining algorithms may require a lot of computational resources and run several hours or even days, depending on the dataset and the parametrization of the task.

The next step should require the user to upload a transaction dataset meeting some predefined format [R1]. This dataset should be filtered and transformed to meet the models defined in the backend of the application [R2]. The user should then be given the option to select all data which will be used in the mining process [R3][R4].

In the next step of the pipeline, the user shall create some mining algorithms, which each may run in a separate thread [R5][R22][R25][R26]. Each of these mining tasks shall have a default setup, but may be parametrized by the user if needed [R6][R21]. The application shall run the mining algorithms on the selected data and give feedback about the current status, success, and failure [R7][R8][R23].

If the algorithm finishes successfully it shall produce a result file with information about the frequent patterns in the dataset [R20]. All results shall be collected in a single table [R9]. Each row in this table shall be the frequent pattern, the columns shall be gender-age-groups and the cells shall display the support values. The table rows shall be sorted according to the highest support values in each row. The user shall be able to filter this results table to find more meaningful and interesting patterns [R10]. Further, the user should be given the option to export the table to a downloadable file [R11].

3.3 Exploit of Frequent Disease Patterns

The table of the frequent patterns found in the sequence dataset may provide a summarized visualization of the mining tasks. Differences regarding gender- or age-groups could be observed just by this table. However, the pattern itself may not be useful without the knowledge about the codes it was composed of. To extract and exploit information about a specific frequent pattern the application shall find the

complete set of patients having that pattern as a sub-sequence [R12]. This set shall be used to detect the most frequent diseases in that group [R13]. Further, this set shall be used to build an interactive model of the visualized patient disease trajectories [R14]. The user shall be able to select different gender- or age-groups and filter the model for diseases and disease-groups [R15]. The model shall display support values for the transitions between different conditions and visualize them in a meaningful way [R14]. The user should be given the option to export the model to a downloadable file [R16].

4. DESIGN

The design chapter is built upon the requirements to define the overall system architecture, models and user interfaces that are used for implementation. The development was an incremental process. Experience gained during the implementation and expert knowledge influenced the design.

4.1 ARCHITECTURAL DESIGN

The architectural concept is the first step to build the intended framework to fulfill the defined requirements. Since the Disease Pattern Miner is not integrated into an existing system, a monolithic design was chosen to implement the application as a single contained unit having several modules. The modules of the framework, as illustrated in Figure 7, are explained in more detail in the following section. Note that, essential classes, interfaces, and data-structs of some modules are explained later in Section 5.3.

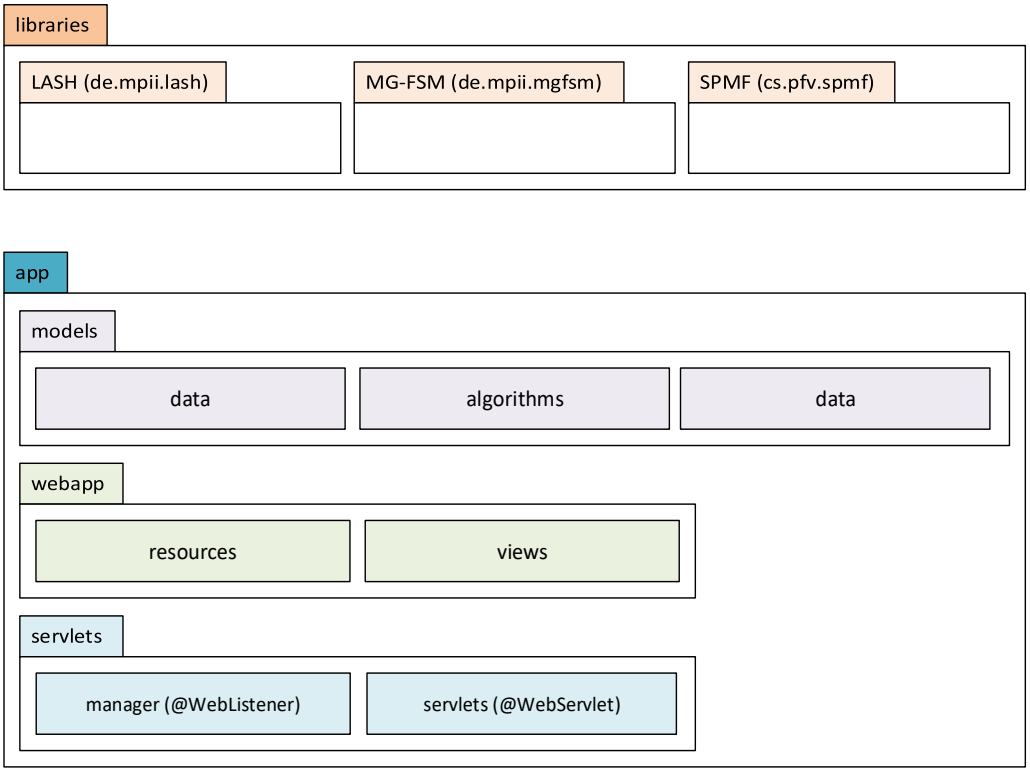


Figure 7. Framework module design.

The Disease Pattern Miner consists of two main modules – `libraries` and `app`. The `libraries` module includes existing implementation of mining algorithms as

independent child modules, e.g., `SPMF`[1], `LASH`[2], `MG-FSM`[3]. It is thus ensured that the framework can be extended by other algorithms or libraries. Since the SPMF library is developed in Java and appears to be the largest open-source collection of pattern mining algorithms, all included mining algorithms have to be Java implementations [R31].

The main module `app` includes three child modules – `models`, `webapp` and `servlets`. These modules are implemented as part of this work to meet the defined requirements from Section 3.1. The Model-View-Controller architecture is used to achieve a separation of concern, which is also implied by the Servlet and JSP Web technology. The module `models` includes three packages – `data`, `algorithms`, and `results`. They form backend models and represent the state of the application. The view is implemented by the module `webapp`, which has the package's `resources`, containing static resources (e.g., CSS, JavaScript, Images), and `views`, comprising dynamically generated web pages. Communication between model and view is realized through the `servlets` module. The `servlets` are divided between either being `managers` (implementing `WebListeners`) or `servlets` (implementing `WebServlets`). The `managers` are started up when the application is loaded into the context while the `servlets` are invoked by an user action.

4.2 User Interface Design

Based on the Disease Pattern Miner requirements and the defined mining pipeline several low-fidelity mock-ups were designed with prototyping tool Webflow [50]. The website user interface follows a minimalistic design principle around the content and main functionality. This ensures that the user can identify the goals and navigate the application step by step. The following section introduces mock-ups to show different visual aspects of this work. The mock-ups are later used to design the Web application.

Figure 8 shows the mock-up of the main template, which is used across the entire application. The proposed sketch is built with a standard website design having a header and navbar at the top, a content section as the main body and a footer. This

[1] http://www.philippe-fournier-viger.com/spmf/
[2] https://github.com/uma-pi1/lash
[3] https://github.com/uma-pi1/mgfsm

simple template design can be used across the entire application by simply replacing the content section with the route specific content. The header and the footer are used to enclose the application and show the site identity as well as useful links. Sitemaps can be quickly accessed through the navbar at all time. The first item in the navbar is a *dashboard* page, which is also the index page of the application. The dashboard page provides some necessary information and main links to the project repository, documentation as well as the related publications.

Figure 8. Mock-up of the main template used for all sites.

Figure 9. Mockup for the data view.

The second navbar item is the *data* page. The mock-up of the data page is shown in Figure 9. This page has two columns, which show the available data sorted by the gender-age-group files. The user can select which of the files should be used during the mining process. Further, the user is given the option to upload additional datasets.

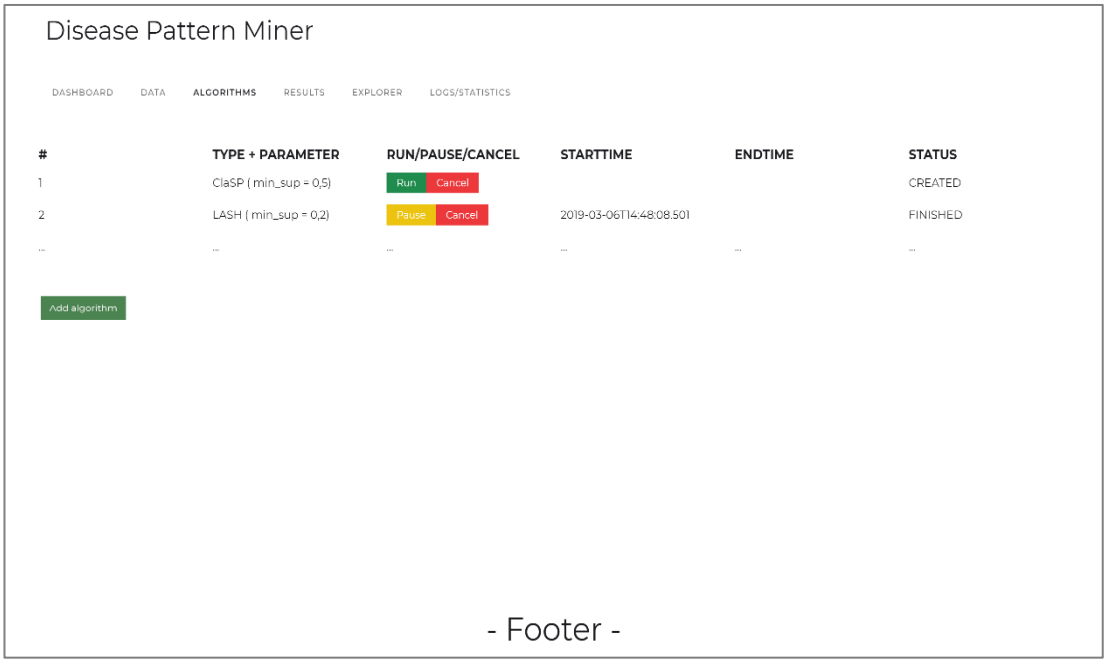


Figure 10. Mock-up of the algorithm view.

The next navbar item is the *algorithms* page. The corresponding mock-up is shown in Figure 10. This page displays a table of different algorithms. The user can add new algorithms, change the type and the parameters. The algorithms can be run, paused and canceled. The table also displays information about status and time when the algorithm was executed and when it finished.

The next item in the navigation bar is the *results* page. The mock-up of the results is presented in Figure 11. The results consist of two sections – filter options to reduce the number of patterns displayed to the most relevant and the results table. The user can add, parametrize or remove the filter. The content of the results table adjusts automatically to these filters. The results table has frequent patterns as rows and the gender-age-groups as columns. The patterns have some color visualization. The cell entries are the support values of the pattern found by an algorithm in the corresponding gender-age-group dataset. Further, the table is a heatmap, and the individual cells are be highlighted according to their relative support values. By adding a color scheme, the user may quickly identify distributions, importance, and similarities of patterns.

Hovering a cell provides additional information about the associated dataset and pattern.

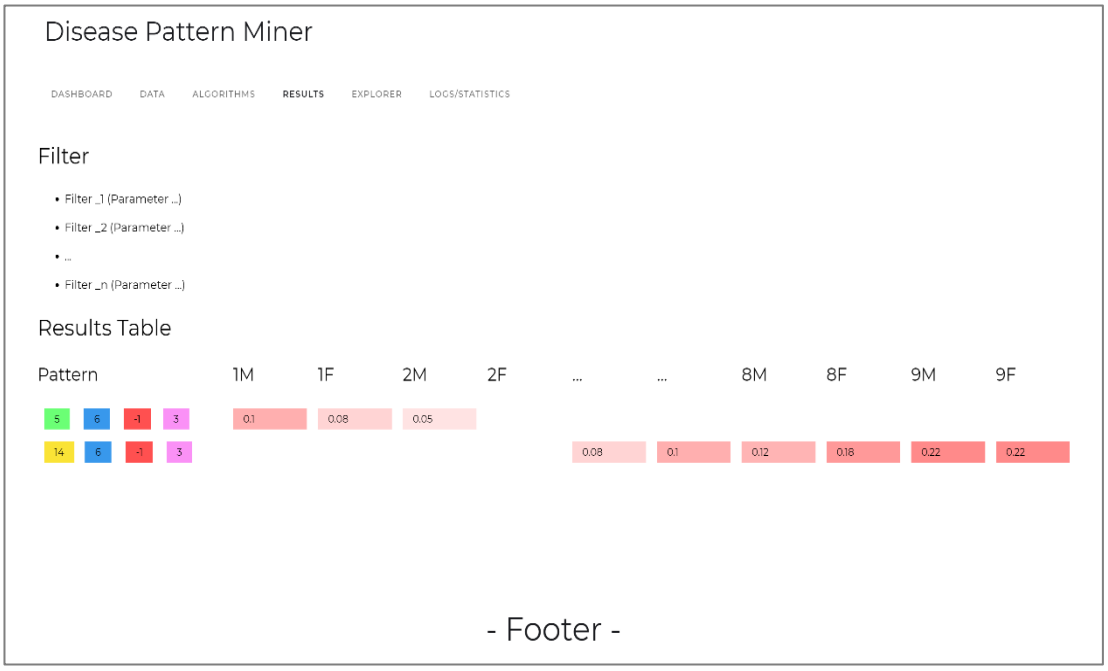


Figure 11. Mock-up of the results view.

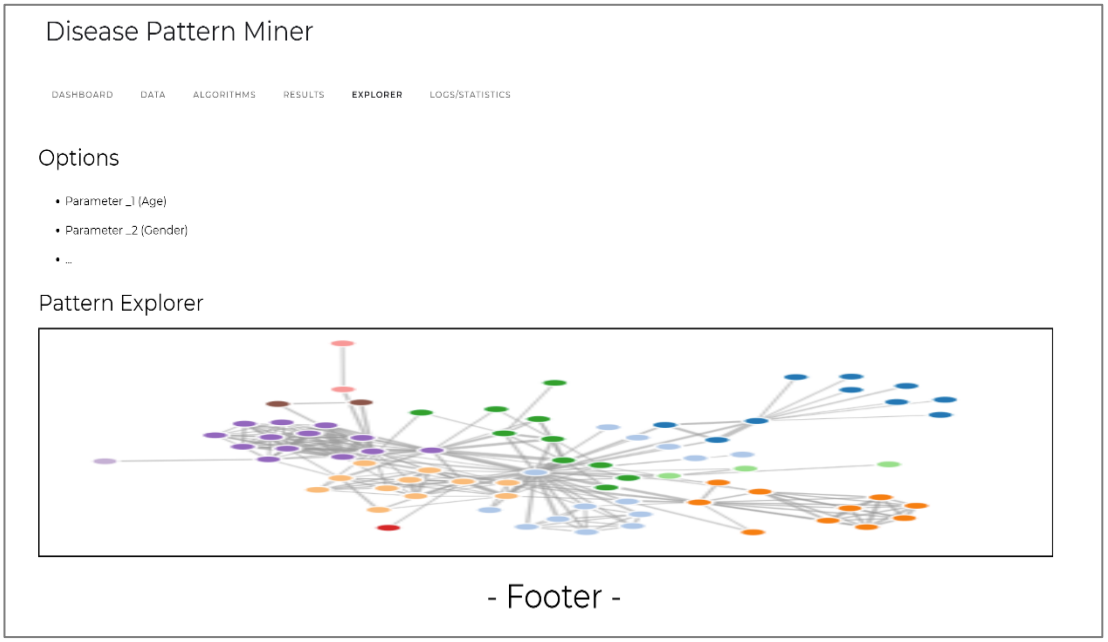


Figure 12. Mock-up of the explorer view.

The next navbar item is the *explorer* page. The mock-up design of this page is shown in Figure 12. The results view is created to compare the found frequent patterns, whereas the explorer view is designed to give insights to a specific pattern selected by the user. A cell or pattern in the results view is linked to the corresponding explorer

view. The explorer view has some basic options to adjust the model, e.g., options to change gender or age. The main content section displays some interactive model build from the set of patients having a specific pattern. The model visualizes patient disease trajectories and displays the frequencies of disease sequences. The user can select specific trajectories to highlight their connections and frequencies.

The last item in the navbar is a *logs* or *statistics* page. This view is not a main component of the framework, and therefore no particular mock-up is designed. This view provides statistical information about the algorithm runtimes and displays them as text or graphs.

5. IMPLEMENTATION

The development of the Disease Pattern Miner followed an agile approach with multiple iterations for the application features. The different modules, as well as the frontend, were created according to their design purpose. This section gives an overview of software tools, the included libraries, and models used during the implementation stage.

5.1 SOFTWARE DEVELOPMENT AND THIRD PARTY LIBRARIES

The development of large applications requires different tools to manage the project, e.g., language version, editor, Web framework. This section introduces the software tools and third-party libraries, which were used to develop the disease mining framework described in this thesis.

The application was mainly developed in Java. Therefore the integrated development environment (IDE) *JetBrains IntelliJ IDEA* [51] was used. *IntelliJ* comes with *Git* [52] and *Maven* [53] integration out of the box and offers a user-friendly interface. Further, IntelliJ can support nearly any language through the use of custom language plugins.

Git is a distributed version control and source code management system. During the development, a remote *GitLab* repository was used. After the implementation was finalized, the project was moved to a public *GitHub* repository[4] and published open source under the MIT License to allow further research and the continuation of the development [54].

Maven is used as a software project management and comprehension tool. *Maven* allows dependency management including automatic updating and dependency closures. The main modules `libraries` and `app`, as well as the child modules in `libraries` each, come with their *.pom*-file. This allows each module to compile, test and build independently while having a hierarchical dependency within the application.

[4] https://github.com/vitaliy-ostapchuk93/disease-pattern-miner

The framework uses several third-party libraries. Only the most relevant libraries for this work are introduced in the following section. Dependencies of the `algorithms` child modules are not included. The complete dependency graph of the project is provided in the public repository on GitHub [54].

The main dependencies used for the Web application development are the Java™ EE Specification API's [47], including Servlet, JSP, JSTL and more. This library mainly is used to implement the communication between client and server (see Sections 0 and 4.1).

Another important dependency is the *Apache Commons IO* [55] library. It contains many utility classes, stream implementations, file filters, file comparators and more. This library is used to perform fast read and write operations on the data.

One of the core dependencies is the *Guava* [56] library. *Guava* contains several different libraries developed by *Google*. The most important used during the implementation of this framework is the *Guava's* `Table` interface and its multiple implementations. These collections are used to implement the results table including sorting and various filter options.

To provide some additional useful features to the Java 8 streams the library *StreamEx* [57] is used. Streams help process large collections of data without keeping all in memory at once.

To design a responsive, modern and modular user interface the CSS framework *Bulma* [48] is used. *Bulma* is based on *Flexbox* and offers a large amount of well-documented elements and design options, e.g., layouts, forms, buttons, modifiers, containers, icons.

The application uses JavaScript on the client side to perform some HTML manipulations, event handling or CSS animations. For example, the library *Tippy.js* [58] is used for the tooltips animation and *jQuery* [49] used for asynchronous requests to the server.

To visualize some of the data the *Plotly [59]* library is used. *Plotly* is a free, open-source chart visualization library available in both JavaScript and Python.

Another important dependency to mention is the Data-Driven Documents (D3) JavaScript library [60]. D3 is a framework build for visualizing data with Web

standards like HTML, SVG, and CSS. This library was used to build an interactive model of the disease sequences. To be more specific, a Sankey diagram [61] with the extension of hierarchical bi-directional flow [62] was adapted to fit the requirements of the model.

5.2 DATA PROCESSING PIPELINE

The sequence mining algorithms use file references as input in the mining process, and the given dataset is provided as a single *CSV*-file. Therefore, the application is using the file system storage instead of a database for all data manipulations.

The dataset used in this study (see Section 2.6) is provided in the form of a *CSV*-file. Each line is a transaction which encodes the gender, age-group, patient identifier, date and one to three ICD-9-CM codes. Transactions which do not match the given pattern or encode invalid values are removed from the dataset before further manipulations. An example snippet is shown in Table VIII and is used in the following section to illustrate the filtering and transformations to create the disease groups sequence datasets. The resulting filtered and encoded sequence datasets are shown in Table VIII.

Table VIII. Example snippet from the NHIRD dataset.

TID	Gender [FEMALE/MALE]	Age [0-9]	Patient-ID	Date [YYYYMMDD]	ICD-9-CM Codes
1	FEMALE	0	PV63270480	20010108	{410}
2	MALE	3	KT61521480	20010107	{0740, 4661}
3	MALE	3	KT61521480	20010107	{V202}
4	FEMALE	0	PV63270480	20010109	{4659,7806,465}
5	FEMALE	0	PV63270480	20010127	{460,94400}

Several transformations and filters are applied before performing a mining process on the data. Figure 13 illustrates the different steps in the data processing pipeline. First, by splitting the dataset in gender-age-group *CSV*-files, the mining task can be performed in specific groups of interest while also reducing the base size of the input file to the base size of the specific group file. For instance, the example dataset from Table VIII is separated into two files corresponding to the gender-age-groups $F0$ and $M3$. Since all transactions in a group file have the same gender and age-group, these rows are dropped. The remaining dataset grouped by the patient identifiers and

sorted by the date. Based on expert advice, ICD-9-CM codes of each transaction are reduced to their category (first three characters). The reason for this is to reduce the code to the most general part and not include too specific diagnoses. Duplicate or invalid codes in a transaction are removed. For example, the transaction from Table VIII with $TID = 4$ is reduced to the codes $\{465,780,465\}$ and the duplicate category code 465 is removed, which leaves the transaction code set $\{465,780\}$. Based on expert advice, ICD-9-CM codes 460 (*common cold*), 760 to 779 (chapter "*Certain Conditions Originating In The Perinatal Period"*), V01 to V91 (chapter "*Supplementary Classification Of Factors Influencing Health Status And Contact With Health Services"*) and E00 to E99 (chapter *"Supplementary Classification Of External Causes Of Injury And Poisoning"*) are removed. This filter operations are performed to remove the unreliable, inaccurate or non-relevant parts of the data. Empty transactions are removed after the filter operations. The patient ID is used as the SID of the corresponding sequence. These transformations result in a disease category sequence dataset (see Table IX).

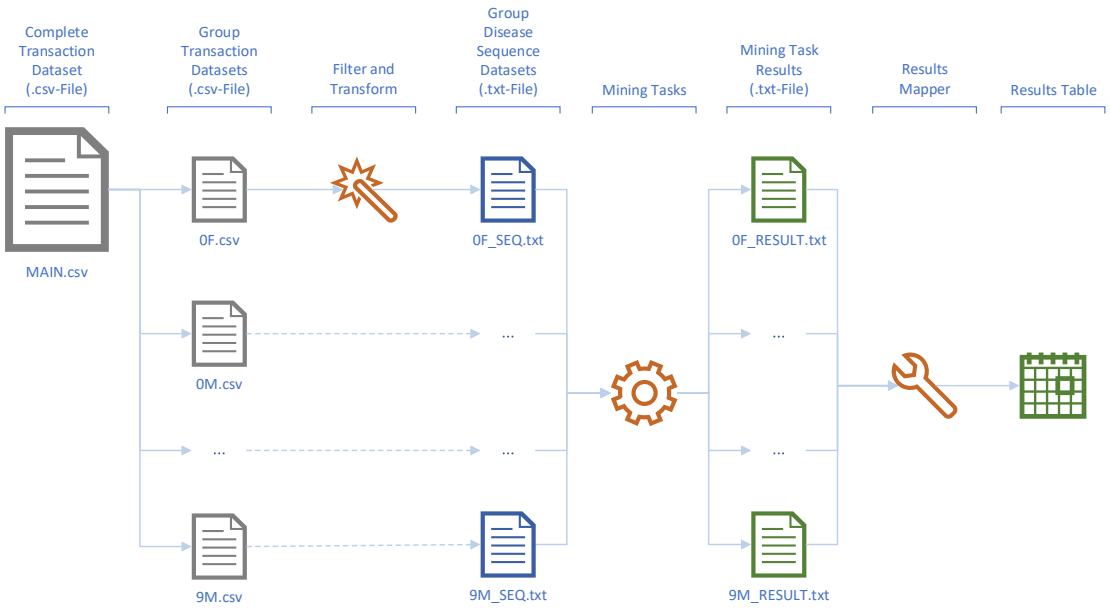


Figure 13. The data processing pipeline.

Based on expert advice, transactions are merged if their dates are less than two weeks apart. This time gap was chosen to join disease codes, which usually belong to the same treatment period. Duplicate category codes within the resulting transactions are removed. For example, transactions shown in Table VIII with the *TID* one and four, are recorded only one day apart and are therefore merged to $\{410,465,780\}$. Each

disease category in the sequence is encoded with the corresponding chapter of the ICD-9-CM codes, e.g., 465 corresponds to *"Diseases Of The Respiratory System*". This generalization was made intentionally to exploit the hierarchical concept of the ICD encoding and reduce the number of distinct items. An enumeration is used to represent each chapter. Duplicate enumerations within a transaction are removed. Sequences of length less than three or less than two distinct enumerations are removed. For instance, the disease category sequence with $SID = KT61521480$ in the $M3$ dataset (see Table VIII) does not have a valid encoded sequence, since the item $V20$ was filtered and only two valid items remained. Items within a transaction are sorted according to their ordinal values of the enumeration, e.g., *"Diseases Of The Respiratory System*" corresponds to the ordinal seven of the disease chapter enumeration. To match the *SPMF format*, the chapters are encoded as the corresponding ordinal values separated by single space. The end of each transaction is encoded as "-1" and the end of the sequence is encoded as "-2", i.e., the sequence with $SID = PV63270480$ in the $F0$ dataset from Table IX would be encoded as "$6\ 7\ 15\ -1\ 16\ -1\ -2$".

Table IX. Example of the sequence dataset from Table VIII.

Gender-Age-Group [F/M + 0-9]	**SID [Patient-ID]**	**Disease Category Sequence**	**Encoded Sequence [SPMF]**
F0	PV63270480	$\langle\{410\},\{465,780\},\{944\}\rangle$	$6\ 7\ 15\ -1\ 16\ -1\ -2$
M3	KT61521480	$\langle\{074,466\}\rangle$	

Each of the valid SPMF encoded disease sequences of a transaction gender-age-group *CSV*-file is written as a line to a corresponding sequence *txt*-file. This data transformation is performed by the system automatically when the application deployed on the server or the user uploads new data. In the data view, the user can select which sequence files are used as input for the mining algorithms. The output of each successful mining task is a *txt*-file in SPMF format. Each line in the result file is a frequent sequential pattern. At the end of each line the keyword "$\#SUP:$" followed by an integer indicating the absolute support of the pattern as a number of sequences, e.g. "$4\ 2 -1\ 5 -1 -2\ \#SUP: 6$" would be a frequent sequential patter consisting of the itemsets $\{4, 2\}$ followed by the itemset $\{5\}$ and having the absolute support $\sigma = 6$.

5.3 Relevant Classes

This section lists the most relevant classes and their dependencies. Low-level details or code examples are not included, but non-trivial implementation issues and concepts are explained.

`AbstractAlgorithmTask`	Implements the concept of a mining task, which is inherited by specific mining algorithms, e.g., `BidePlusTask`, `SpamTask`.
`AlgorithmManager`	Starts when the servlet context is initialized. Uses an `ExecutorService` as a thread pool for the `AlgorithmRunnables`.
`AlgorithmRunnable`	Implements a `Runnable` to enclose a mining task as a separate managed thread.
`DataManager`	Starts when the servlet context is initialized. Loads all datasets and references into memory to create the state of the application.
`DiagnosesGroupHelper`	Implements most of the ICD-9-CM encoding data manipulations in the application, e.g., chapter (`DiagnosesGroup`) or code (`ICDCode`) filter, get proper description or name of the `ICDCode`, get the color to highlight an `ICDCode` or `DiagnosesGroup`.
`ICDLink`	Implements a concept, which is used to build a model of patients' disease sequences. The `ICDLink` is defined by a source node, a target node, and a link value. Given a disease category sequence of a patient, the `ICDLinks` are built of `ICDCodes` appearing in sequential order. The value represents the frequency of the two codes in the given sequence.
`ICDSequence`	Implements the concept of a disease category sequence of a patient. This class is used to compare itemsets or patients and built an encoded sequence dataset. Further, this class is used to analyze the patient disease trajectory to build an `ICDLink` collection.
`InverseSearchThread`	Implements a `Thread`, which uses the `PatternScanner`, to perform an inverse search, which retrieves all patients, who contain a specific frequent pattern, for all frequent patterns in the `ResultsTable`.
`PatternScanner`	Implements the concept of an inverse search, which retrieves all patients, who contain a specific frequent pattern. This subset is used to create `ICDLinks` and extract the most frequent `ICDCodes`. This operation may be

expensive in terms of memory and runtime. Therefore, it is performed only once and the data is stored in *JSON*-files.

`ResultsMapper`

The `ResultsMapper` collects the frequent patterns and the corresponding support values of all result files in a single `ResultsTable` collection, which is presented to the user in the results view.

5.4 Deployment and Maintenance

The Disease Pattern Miner is built and deployed on both local and remote development environments. Both use the Apache Tomcat [63] servlet container (version 9.0.12) to deploy the application as a Web service. The Tomcat server may monitor metrics and runtime characteristics of the application. Additionally, a Jenkins [64] server (version 2.138.2) is set up on the remote to support continuous integration and delivery process for this project. In Jenkins, a pipeline is configured to trigger whenever new code is pushed to the remote GitHub repository of the project. Jenkins automatically pulls the most recent code from the remote repository, builds the application with Maven, according to the *pom*-files specifications, and may execute some tests. If the build is successful Jenkins deploys the application *war*-file to the Tomcat server. This setup helps to reduce cost, time and risk of delivering changes and allowing incremental updates to the application in production.

6. RESULTS

This chapter presents the developed Disease Pattern Miner. This includes the user interface of the Web pages as well as the results of the mining tasks and the modeling of the frequent disease patterns.

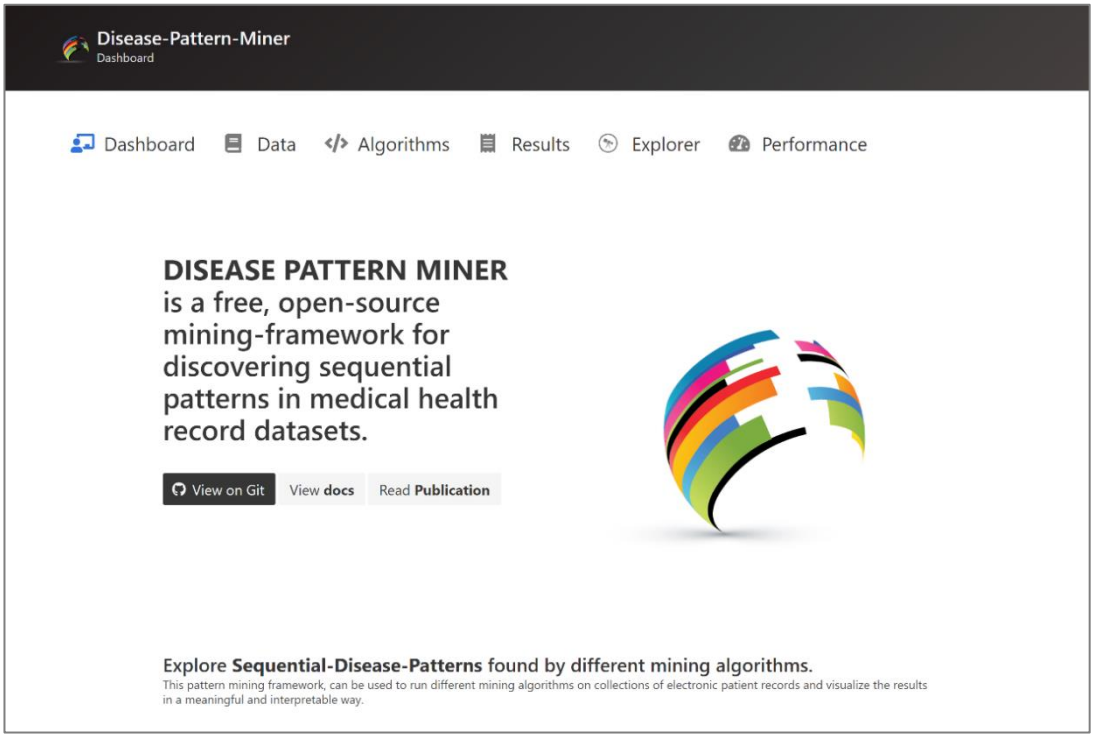


Figure 14. Final implemented the interface of the dashboard view.

6.1 PATTERN MINING FRAMEWORK

The Disease Pattern Miner is developed as a monolithic application which is deployed on a Tomcat server. The application is developed with modern, responsive Web technologies. The user can access the frontend with any browser. As described in Section 4.2, the application follows a minimalistic, consistent layout having a header, navigation bar, main content section, and a footer. The navigation bar provides internal links to six different views. In each view, the user can interact with the application to identify and archive specific goals or make observations. For each view screenshots of the final implemented interfaces are presented briefly in the following section.

The dashboard, shown in Figure 14, is the index page of the Web application. In this view, the user is presented a short description of the framework and three external

links to the public GitHub repository, documentation and scientific publications related to this project.

The implementation of the data view is shown in Figure 15. In this view, the user can decide which datasets are used in the mining processes. By clicking the gender-age-group files, they can be selected or unselected. In the top part, the user is given two options to add a dataset to the application context. The first option is to upload a new dataset as a single *CSV*-file matching the predefined format. The second option is to download a *zip*-file from an external source, e.g., GoogleDrive. The archive may contain already processed datasets, e.g., result files or inverse search files. This option may be useful to share datasets which are processed by some other user.

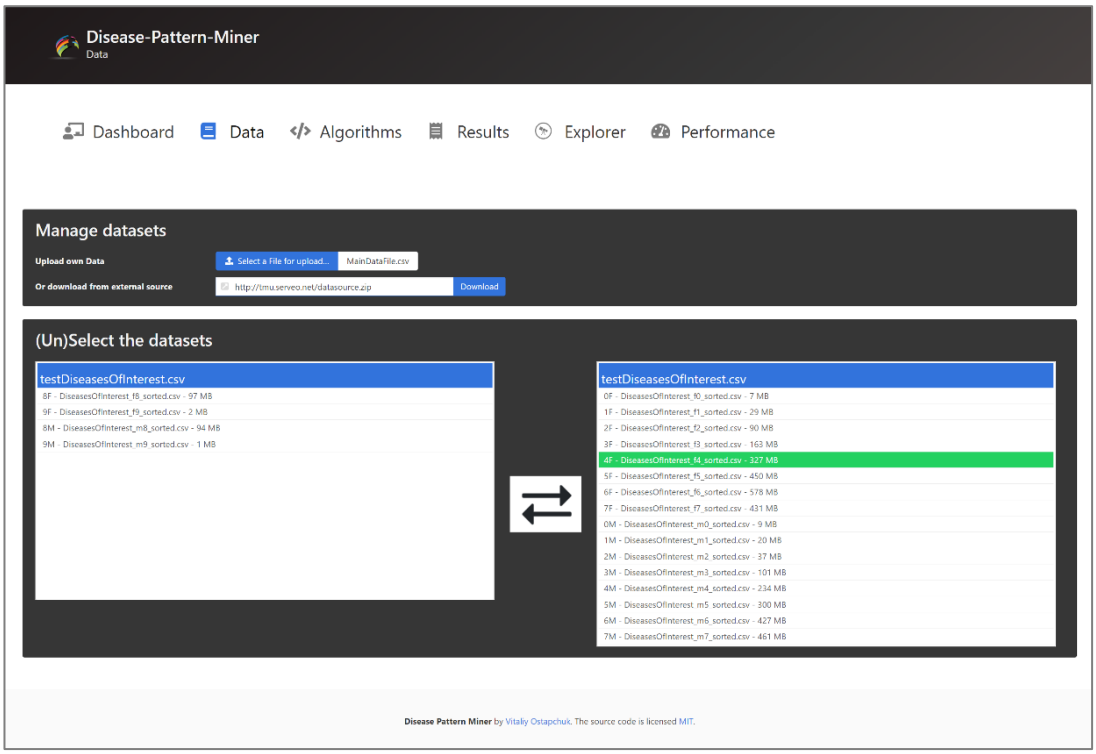


Figure 15. Final implemented the interface of the data view.

Figure 16 shows the implementation of the algorithms view. The user can chose from 15 sequence mining algorithms to create mining tasks, which are displayed as rows in a table. The algorithms and constraints (see Section 2.2.4 and 2.4) have default parameter values, which may be adjusted for each task. The user can start, pause or cancel each mining task. The mining tasks are run as separate threads on the selected dataset and produce a result. The table shows the current status of the algorithm, time of creation, start and end, as well as the duration. Each task is run as a separate thread in a managed thread pool.

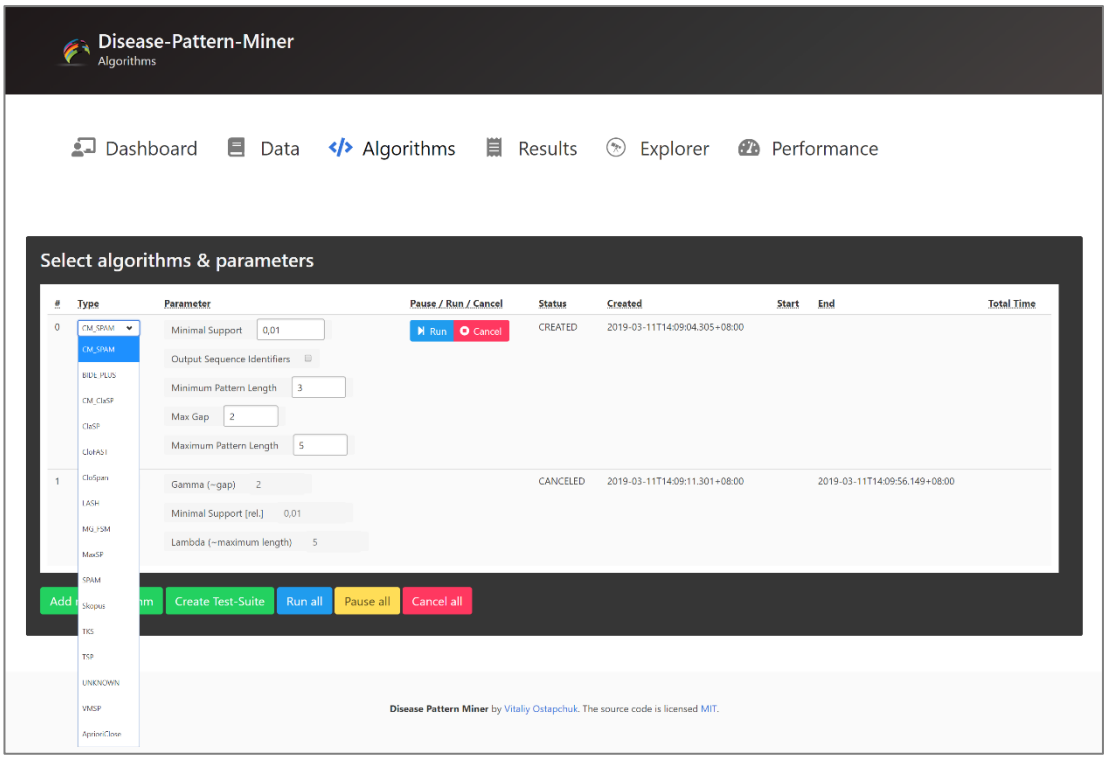


Figure 16. Final implemented the interface of the algorithms view.

6.2 SEQUENTIAL DISEASE PATTERNS

The Disease Pattern Miner automatically filters and converts the uploaded dataset to gender-age-group files having encoded disease sequence information (see Section 5.2). This sequence files of males and females in the age-groups zero to seven are used in multiple mining tasks. The system finishes all tasks successfully and produces result files with information about frequent patterns and the corresponding support values. Additionally, the application logs statistical information regarding the process duration.

6.2.1 Disease Patterns in Results-View

The application collects information from all result files in a single table which is displayed in the results page of the application (see Figure 17). The table shows the found frequent disease patterns as rows and the gender-age-groups as columns. The cell entries of the table show the relative support of the pattern in the corresponding subset. The patterns and the cells are highlighted with colors. All table entries have tooltip information which is displayed on a mouse hover-event.

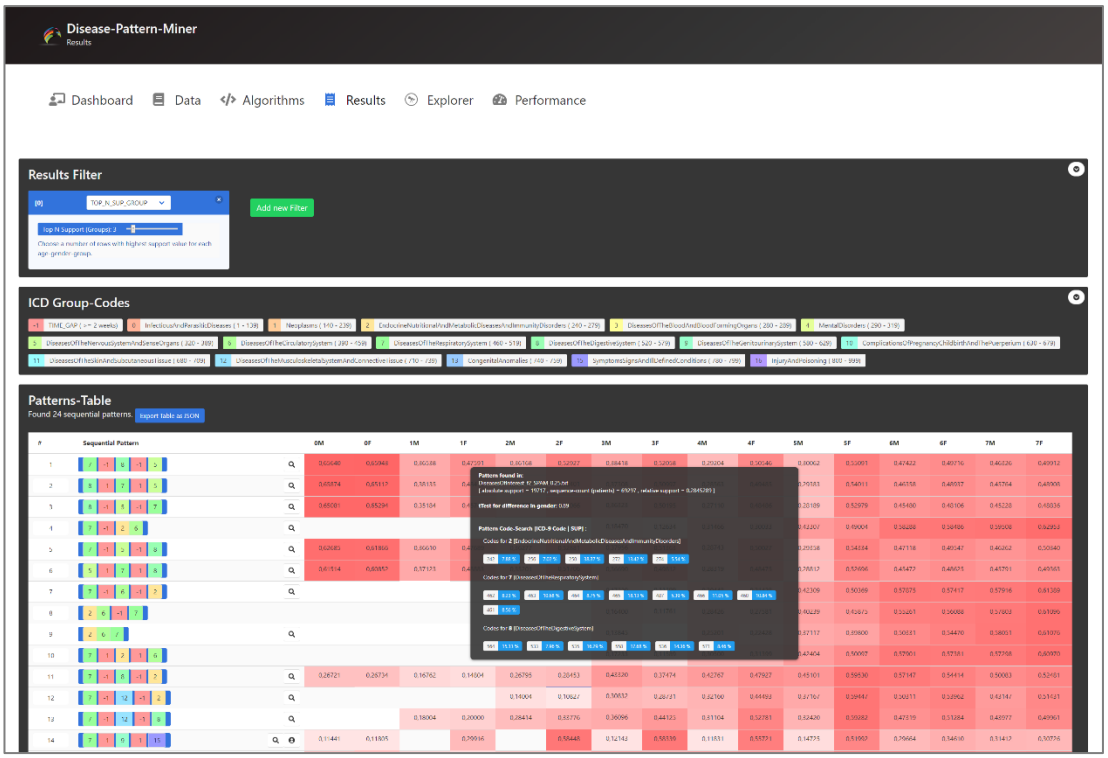


Figure 17. Final implemented the interface of the results view.

The user can filter the table to reduce the number of displayed patterns to a manageable size. The user can select from 11 different filter types, which can be adjusted by changing different parameter values. If multiple filtering options are selected, the filters are processed in order of the occurrence. The application automatically updates the results table. The user can export and download the entries of the filtered results table as a *JSON*-file.

Next to the sequential disease pattern, two icons are displayed. The search (🔍) icon indicates whether the inverse search (see Section 5.3) was already performed for this pattern in all related gender-age-group datasets. The person (👤) icon indicates whether the t-Test is showing a significant difference between males and females when comparing the corresponding relative support values.

Additional information regarding the disease sequence pattern in the proper gender-age-groups dataset is displayed as a tooltip of a nonempty table cell. The tooltip includes information about common codes (see Section 5.3) of this cell entry. The user is also able to click the entry to go to the related disease pattern explorer view.

6.2.2 Disease Patterns in Explorer-View

Given a frequent pattern the Disease Pattern Miner can perform an inverse search to find the subset of patients showing the pattern in the corresponding dataset. The application uses the disease category sequences of the patients in this subset to count the frequencies of categories appearing in sequential order. The information is used to build and display a hierarchical, bi-directional, interactive Sankey diagram (see Section 5.1). This model is presented to the user in the explorer view. A screenshot of the pattern explorer view interface is shown in Figure 18.

In the top part, the application displays the selected pattern, which is used for the current Sankey model. The user can adjust the model by selecting the gender and scroll through age-groups. The model can be exported as an image of the Sankey chart or as a *JSON*-file.

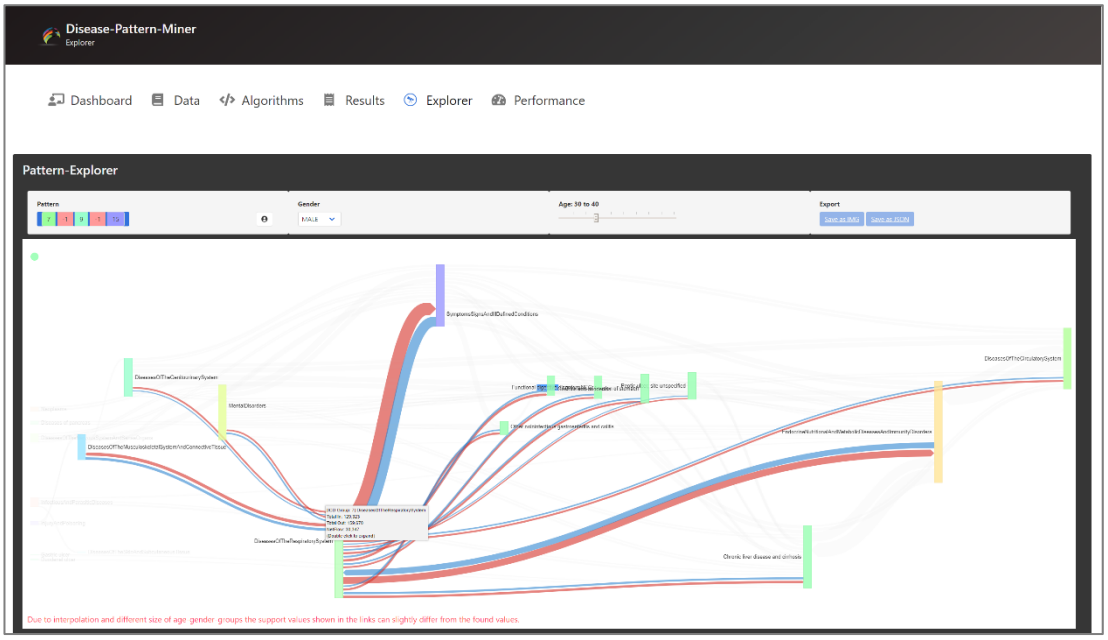


Figure 18. Final implemented the interface of the explorer view.

The Sankey diagram is built of the 250 most frequent disease category links (see Section 5.3). The nodes of the graph represent the different ICD-9-CM disease categories and chapters. The size of the nodes and links give visual feedback about the absolute support values, e.g., large links correspond to high support. The user can move the mouse over the links and nodes to get a tooltip about the proper frequencies and highlight the flow. The nodes may be dragged with the mouse to adjust their position. If the node represents a chapter, it may be expanded to the child disease categories. If the node represents a category, it may be collapsed to hide it in the

diagram. Both operations may be reversed by clicking the corresponding node collapser in the left-top corner of the chart.

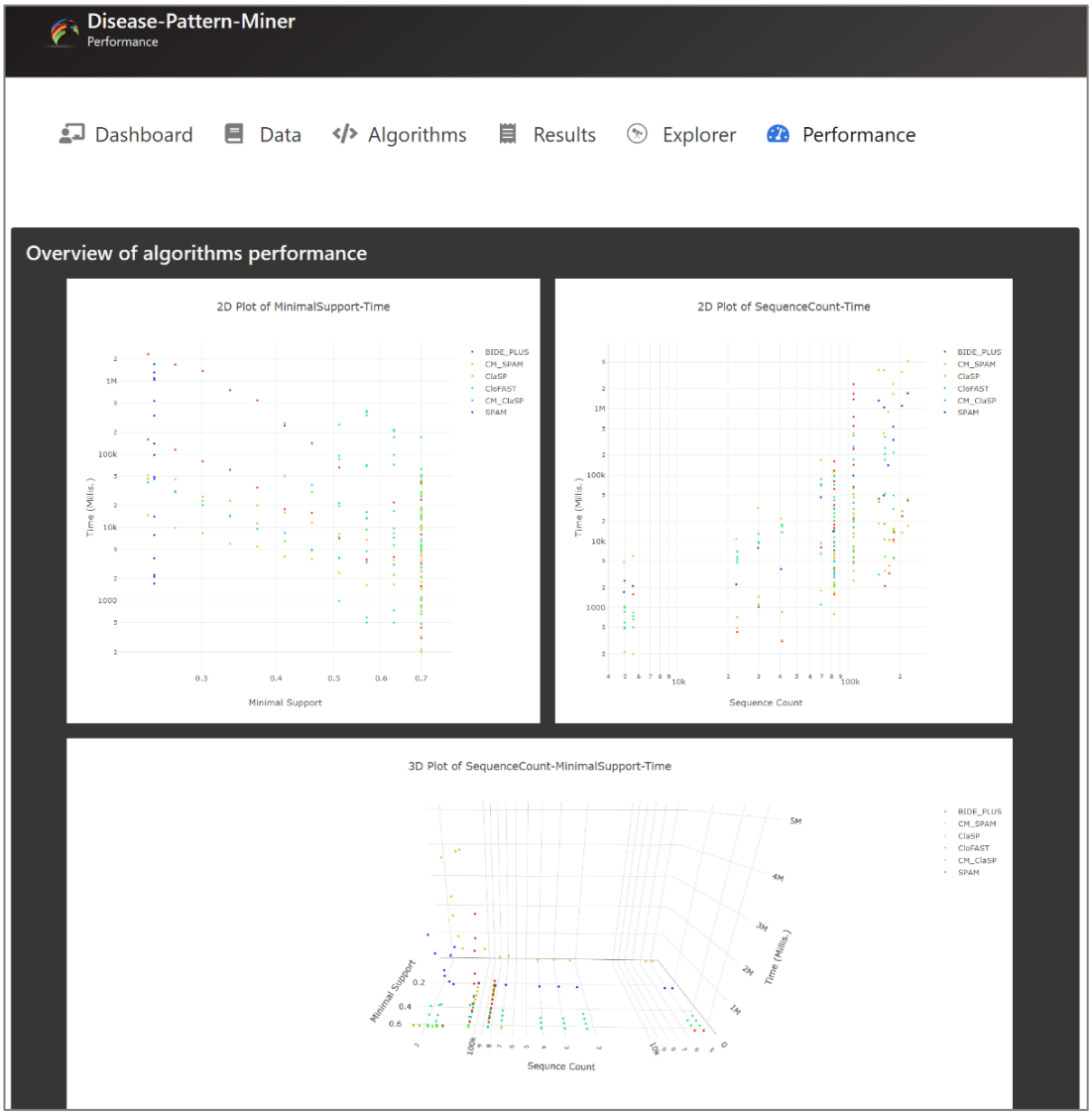


Figure 19. Final implemented the interface of the performance view.

6.3 ALGORITHMS STATISTICS

To give additional feedback about the duration of the different mining tasks the performance interface, shown in Figure 19, was implemented. The Disease Pattern Miner uses data collected from log files to display the duration of different mining algorithms in simple scatter plots with the Plotly library (see Section 5.1). The duration is compared only to the minimal relative support threshold and the size of the dataset. Note that it is not the intention of this thesis to evaluate the performance or state which algorithm performs best. However, this information might help the user to choose algorithms and parameter values for different mining tasks. It is also important to note

that it is problematic to compare the algorithms with regard to the many different parameters and dataset characteristics. Due to these differences, not all of the algorithms' performance values are displayed in the plots.

7. EVALUATION AND DISCUSSION

In this chapter, a reflection of the presented technical solution is given. Further, this chapter presents a stepwise scenario of using the implemented Disease Pattern Miner. For each step, an example analysis of the observations is outlined.

7.1 ANALYSIS OF THE PATTERN MINING FRAMEWORK

Within the scope of this project, a framework for disease pattern mining was designed and implemented. This section will discuss whether the application satisfies the defined requirements (see Section 3.1).

The conceptual framework devised is based on a solid foundation of concepts and algorithms for sequential pattern mining. The key motivation for the development of this domain-specific application is to run many of the proposed algorithms to discover valuable knowledge about frequent disease patterns and disease co-occurrences in EHR datasets. The Disease Pattern Miner meets all statutory requirements and conditions (see Section 3.1). The defined requirements are tested in production by multiple users according to the workflow described in the mining pipeline (see Section 3.2). The functionality was extended to not only find the frequent disease patterns but to build an interactive model, which lets a data scientist explore the disease trajectories. This model may provide valuable insights for further research on disease-disease associations. The framework was published open-source under the MIT License in a public GitHub repository [54].

The application was developed on Windows and later deployed to and hosted on a remote Linux (Ubuntu) environment. The framework appeared the same, except for performance differences which can be attributed to the fact of different capabilities of the processor, RAM, and hard drive. After the implementation, the framework was in production for more than a month to find and fix minor errors. Metrics and runtime characteristics have been collected to show adequate functioning. The session availability showed an average 98.8% per week with a total downtime of 60.48 minutes in the measured period. The downtime can be explained by the server availability and maintenance of the project. The application reloaded the context

successfully in all cases within less than a minute. Except for the inverse search, the application does not show significant request or response delays.

The Disease Pattern Miner is divided into several views, which can be used by the user to interact with the application. The views are built following a minimalistic design allowing the user to identify the goals in each presented interface fast. Nevertheless, to work with the data and different algorithms requires expert domain knowledge. New data can be uploaded and included in the mining process but has to match the predefined format. The framework does not support other formats or connections to a database. The application can filter and convert the data to match several different formats. Mining tasks with 15 different sequence mining algorithms can be defined. All mining tasks are enclosed in separate threads and can be parametrized. Prior knowledge to the selected datasets is required to adjust the parameters to get the best possible performance and results. The feedback of the tasks is limited to a status report. A result file with frequent patterns is produced if the mining task is successful. The results are collected and presented to the user in a single table. To provide better visual feedback the frequent disease patterns and the support values are highlighted with colors. The user can apply 11 different filter options to find more relevant disease patterns.

For each frequent pattern, the application can find a subset of patients and build an interactive model of disease trajectories. The inverse search may be expensive in terms of memory and runtime but is only performed once. The model is built with a bi-directional, hierarchical Sankey diagram. Both, the results table and the Sankey chart, can be exported as *JSON*-files. The model may be adjusted by the user to show specific gender- and age-groups data. The diagram dynamically renders the new positions of the model objects according to the disease links in the proper dataset. The application does not support to save the state of the model in the application, but the chart may be exported as an image.

The Disease Pattern Miner is developed as a Web application using the Java™ EE Specification API's [47], e.g., Servlets, JSP. These server-side technologies are used for Java application development and offer benefits for monolithic applications based on the Model-View-Controller architecture. Nevertheless, there exist various ways to develop a modern Web application. The modular architecture makes most of the productive code reusable for future development. Dividing the application into a

frontend and backend service, e.g., frameworks like Vue [65] and Express [66], might offer a better separation between the application concepts. Further, to interact with other applications the backend endpoints should be defined in a clear API, e.g., REST or GraphQL. Container-based microservices (e.g., Docker [67]) may offer opportunities to replace the thread-based implementation of the mining tasks.

7.2 Analysis of Disease Patterns

For demonstration and evaluation purposes a subset of interest was selected from the complete dataset. This subset contains only patients who show at least one of the following categories within their disease category sequence: *migraine* (code 346), *diabetes mellitus* (codes 249 and 250), *myocardial infarction* (codes 410 and 412) or *chronic kidney disease* (code 585). Detailed statistics of this subset of interest are shown in Table X in Appendix A.

The dataset of interest was uploaded through the data interface (see Figure 15) of the Disease Pattern Miner. After the upload was successful, the application divided the dataset into gender-age-groups files. The data of age-groups eight and nine tends to have less distinct patients and codes, while at the same time the maximum and average length of the encoded sequences are high. This observation may be explained by the fact, that chronic diseases such as heart disease and diabetes are common among older adults. The encoded sequences of these groups show periodic patterns [68]. Mining periodic patterns is an own field in data mining. The introduced sequence mining algorithms do not cover the concept of periodic pattern mining. Therefore these groups are not selected for the mining tasks.

In the algorithms view, all 15 algorithms were set up for mining with different parameters. Across all mining tasks: the relative support was set 0.1 or higher, the gap constraint (see Section 2.2.4) was set to be two, the minimal sequence length was set to be three and the maximum sequence length was set to be 10. All tasks have been started independently and finished the mining successfully.

7.2.1 Analysis of the Result-View

The Disease Pattern Miner has automatically collected the data from all result files. The mining tasks have been able to find 313558 frequent sequential patterns in the subset of interest. All frequent patterns were loaded into the results view (see Section 6.2.1) automatically.

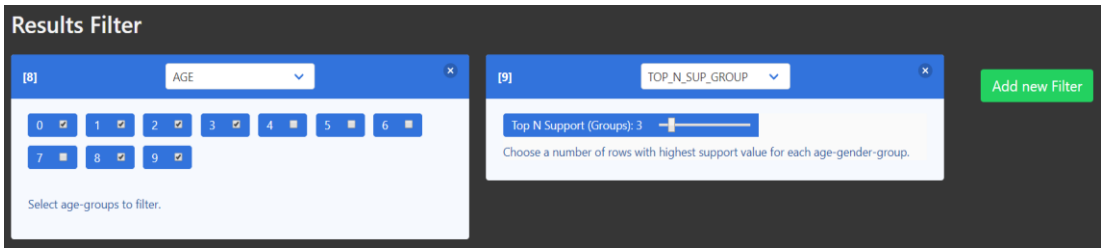
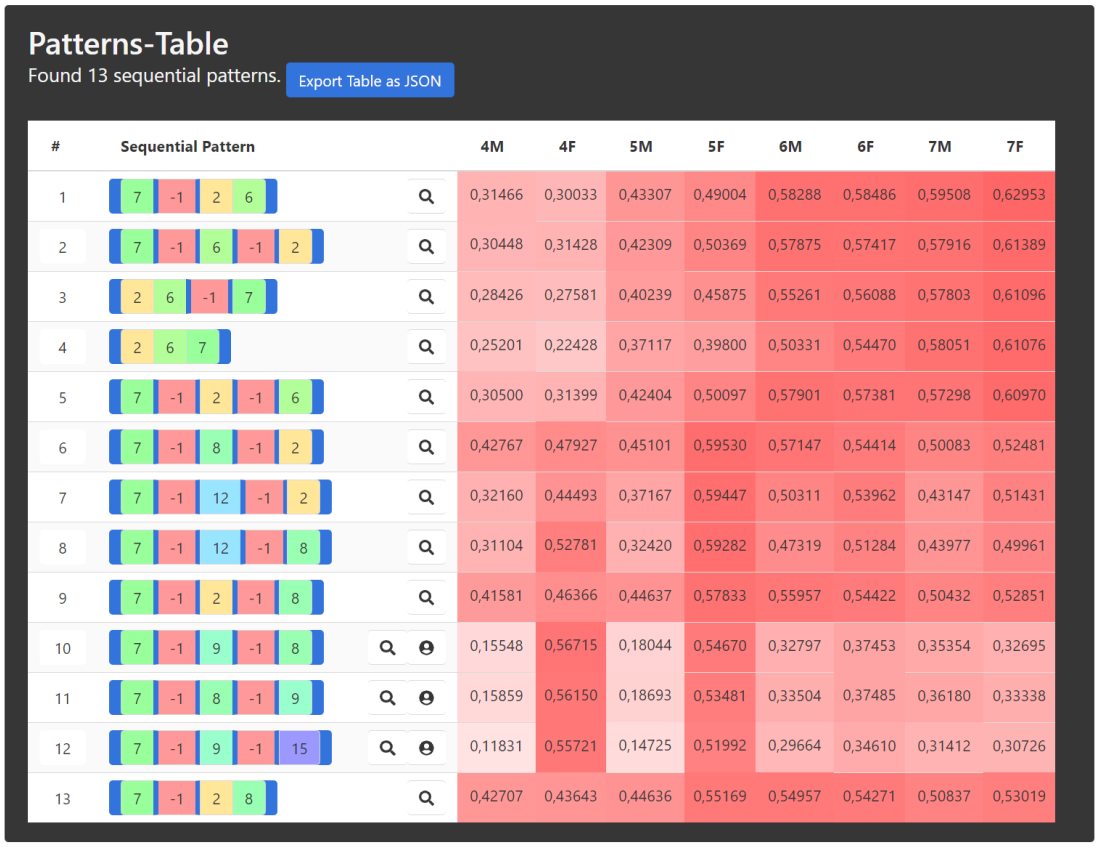


Figure 20. Example of the possible filter operations.

To demonstrate the filter operations and limit the results to a manageable size two filters were created (shown in Figure 20). The *Age*-filter, for the groups zero to three, is selected to display only the groups which have more than 150000 valid encoded sequences (see Table X in Appendix A). The *TopNGroups*-filter is selected to filter the frequent patterns for three patterns having the highest support value in each age-gender-group. Figure 21 shows the pattern table in the results view after the two filter operations are applied.



#	Sequential Pattern	4M	4F	5M	5F	6M	6F	7M	7F
1	7 -1 2 6	0,31466	0,30033	0,43307	0,49004	0,58288	0,58486	0,59508	0,62953
2	7 -1 6 -1 2	0,30448	0,31428	0,42309	0,50369	0,57875	0,57417	0,57916	0,61389
3	2 6 -1 7	0,28426	0,27581	0,40239	0,45875	0,55261	0,56088	0,57803	0,61096
4	2 6 7	0,25201	0,22428	0,37117	0,39800	0,50331	0,54470	0,58051	0,61076
5	7 -1 2 -1 6	0,30500	0,31399	0,42404	0,50097	0,57901	0,57381	0,57298	0,60970
6	7 -1 8 -1 2	0,42767	0,47927	0,45101	0,59530	0,57147	0,54414	0,50083	0,52481
7	7 -1 12 -1 2	0,32160	0,44493	0,37167	0,59447	0,50311	0,53962	0,43147	0,51431
8	7 -1 12 -1 8	0,31104	0,52781	0,32420	0,59282	0,47319	0,51284	0,43977	0,49961
9	7 -1 2 -1 8	0,41581	0,46366	0,44637	0,57833	0,55957	0,54422	0,50432	0,52851
10	7 -1 9 -1 8	0,15548	0,56715	0,18044	0,54670	0,32797	0,37453	0,35354	0,32695
11	7 -1 8 -1 9	0,15859	0,56150	0,18693	0,53481	0,33504	0,37485	0,36180	0,33338
12	7 -1 9 -1 15	0,11831	0,55721	0,14725	0,51992	0,29664	0,34610	0,31412	0,30726
13	7 -1 2 8	0,42707	0,43643	0,44636	0,55169	0,54957	0,54271	0,50837	0,53019

Figure 21. Example analysis in the Pattern-View.

In the following section, an example analysis is outlined. The two filter operations reduced the number of frequent patterns to 13 with the highest support values in each group. The *TopNGroups*-filter should show three patterns for each of

the eight groups (3 patterns in each of 8 groups → 24 patterns). Therefore, it can be concluded, that most of these patterns have the highest relative support in multiple age-gender-groups (24 expected – 13 displayed → 11 patterns have highest support in multiple groups). The heatmap on the right side is indicating that most entries show high support. It can be observed that older groups tend to be highlighted somewhat stronger. A reasonable explanation would be that older people become more susceptible to different diseases. As shown in Table X in Appendix A, this observation is also supported by the average and maximum length of the encoded sequences in these groups, which are higher than in younger age-groups. On the left side, the frequent sequential patterns are displayed. It can be observed that all patterns contain the encoded chapter *7 (Diseases Of The Respiratory System)*, and most also contain the encoded chapter *2 (Endocrine Nutritional And Metabolic Diseases And Immunity Disorders)*, or *8 (Diseases Of The Digestive System)*. The reason for this is that these chapters have a high prevalence in the subset of interest. The search (🔍) icon, next to the sequential patterns, indicates that the inverse search (see Section 5.3) was completed for all patterns. Further, the person (👤) icon is indicating that the t-Test is showing a significant difference between males and females in the three patterns. The striped rows of the heatmap for those patterns support this observation.

Figure 22. Example cell hover-tooltip (group 4*F* of the pattern number 12).

For demonstration and evaluation purposes the 12th pattern "*7 (Diseases Of The Respiratory System) – 1 (TIME GAP) 9 (Diseases Of The Genitourinary System) –1 (TIME GAP) 15 (Symptoms Signs And Ill-Defined Conditions)"* is examined more closely. Note that the selected pattern shows a significant difference in gender–groups,

whereas the support for this pattern in the female subsets is higher. To get more information about the codes which result in this pattern the user may hover the mouse over the cells in the heatmap. The tooltip from the hover-event in the group $4F$ is displayed in Figure 22. In the top part of the tooltip, the user can see which mining algorithm found this pattern, information about the absolute support values and the result of the t-Test regarding the difference in gender. In the bottom part of the tooltip, the most frequent ICD-9-CM category codes are listed. For instance, in this group, the pattern could be observed in 102042 different patients and about 24% category codes of the chapter 15 are built by code 780. The user can switch to the corresponding model in the explorer view, by clicking the cell.

7.2.2 Analysis of the Explorer-View

In the explorer view, the user is presented a model of the disease trajectories, which are built from the disease category sequences of the subset of patients containing the pattern. This section will only outline an example in a female group since the pattern was found significantly more often in the female datasets. The primary interest might be to analyze the links between the three chapters of the selected pattern (*Diseases Of The Respiratory System*, *Diseases Of The Genitourinary System* and *Symptoms Signs And Ill-Defined Conditions*). Figure 23 shows the model of the selected pattern in the group *4F*. All chapter nodes which are not part of the selected pattern are expanded, and all nodes which have no links between the chapter nodes of the selected pattern are removed. The chapter nodes of the selected pattern may also be expanded to see the child disease categories.

The exact support values of the links and nodes can be inspected in a tooltip. For instance, Figure 24 shows the hover-event if the user wants to highlight the links to or from the node *Diseases Of The Respiratory System* on the left or get its total support values (Total In: 1193603, Total Out 1535786, Net Flow: 265261).

Using this model might help to conduct further research on disease associations. This might include finding causes and effects of diseases in the graph, predicting outcomes for individual patients, regrouping certain diseases or finding evidence for disease cooccurrences. Nevertheless, before that, a structured evaluation of the model is needed to validate the associations against already well-researched disease comorbidities.

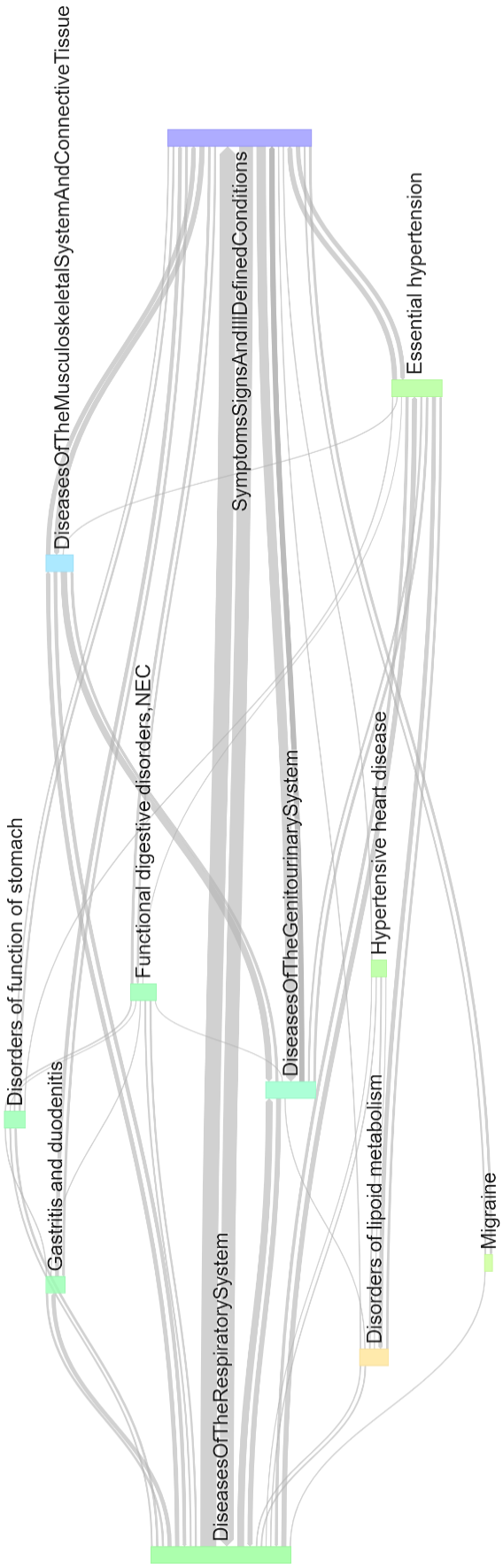


Figure 23. Sankey diagram of the disease trajectories (pattern number 12, group 4*F*).

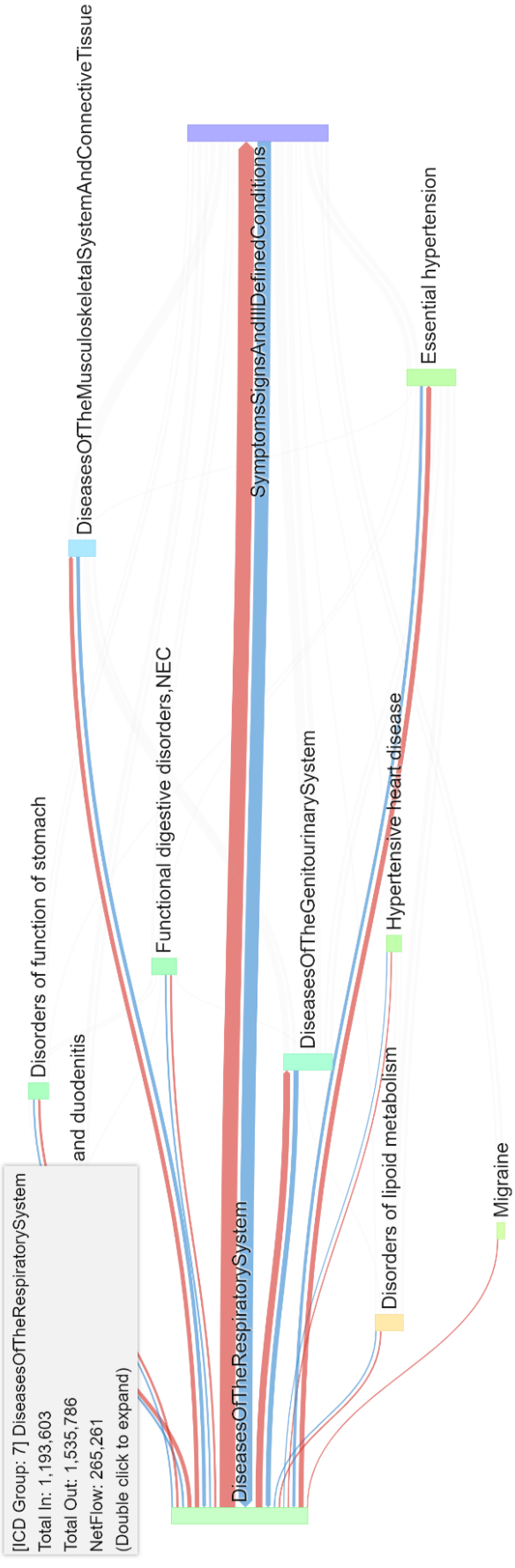


Figure 24. Hover-event for the node *Diseases Of The Respiratory System*.

7.3 Algorithm Runtimes

Note that evaluating or comparing the mining algorithms it is not part of this work, especially since most of them use different constraints and parameters. It could be observed that all algorithms have an acceptable execution time if either the relative support value is high or the dataset is small. However, the execution time and memory of the tasks increases exponentially for both properties. Because of this, the tasks could not complete successfully in most cases if the relative support was chosen low and the dataset was too large.

7.4 Limitations

This section highlights some of the limitations and weaknesses of the Disease Pattern Miner application reported in this thesis.

First and most important weakness is the limited dataset and the many assumptions made during the data processing pipeline. The dataset is almost 20 years old and missing relevant information. For example disease utilities, demographic information like year of birth or the ethnicity, lab results, histological data or information about death are not part of the data. The dataset has to be uploaded by the user instead of establishing a connection to a production database. This might result in corrupt, invalid or duplicate data. The mining is performed after converting the data based on the hierarchy of the ICD-9-CM, which may be considered as a naive approach. The mining process requires expert knowledge about the dataset and the data mining methodology. The mining algorithms only include open source Java implementation. Therefore some important algorithms are missing. In the current application, only sequential pattern mining algorithms are implemented. The mining does not support item or group specific thresholds or parameters.

Although the Disease Pattern Miner satisfies all defined requirements, the monolithic architectural design is not well suited to separate the concepts of frontend and well-defined backend endpoints. Web application technologies are rapidly evolving and may offer great opportunities to improve the design and functionality of the project reported in this thesis. The framework is missing a clear API definition to interact with other applications. Also Unit-Testing most of the current modules is required before deploying the application in production mode.

Other limitations of the work presented in this thesis are some of the results analysis and the model used for the disease trajectories. The property on which the patterns are collected and sorted is the support. Alternative measures of interest, e.g., leverage, may have better properties when it comes to identifying essential patterns. The found patterns have to be reviewed by medical data scientists. The Sankey diagram is useful to represent the flow, but the model elements are re-rendered every time the user changes the age- or gender-group, making it very difficult to determine the effect of individual nodes or links. The model is showing information about the sequential order of two nodes, but the disease trajectory as in the disease category sequence of a patient is not represented.

8. CONCLUSIONS AND FUTURE WORK

Within the scope of this project, a framework for disease pattern mining was designed and implemented to utilize state-of-the-art sequential pattern mining algorithms in a domain-specific Web application. An agile development process was used to develop the Disease Pattern Miner in several stages. In the first stage, basic scenarios and the need for such an application were defined and a requirements analysis was performed (see Section 3.1). Based on the requirements, sequential pattern mining algorithms and libraries have been reviewed and chosen. Further, based on the software requirements specification, a module architecture of a Web application (see Section 4.1) and user interface mock-ups (see Section 4.2) have been designed. In the next stage, the Disease Pattern Miner was implemented (see Chapter 5) and the first results of the mining have been reviewed with domain experts. Changes to the requirements were made, and the functionality was extended to provide an interactive model of the found patterns. The iterative review, maintenance, and improvements of models and the user interface aided the development of the application to a stable release version.

This work has successfully shown that sequential pattern mining algorithms can be embedded in a disease associations data analysis framework. The application may support medical data scientists and researchers to discover valuable knowledge about disease co-occurrences. These insights may help to understand the correlations between diseases or disease-groups, provide retrospective information on a patient trajectory, support prospective therapy decisions, and give epistemological information on diseases which are relevant in a specific gender- or age-groups.

The presented interactive model of patient disease trajectories, which is built from a subset of patients having some frequent pattern, offers a novel approach to find, explore and understand the disease associations. Together with the frequent disease patterns, discovered by various sequence mining algorithms, this application enables a variety of use cases. Nevertheless, a structured evaluation of the data and models is needed.

Future work should separate the application concepts in a frontend and a backend with well-defined endpoints ensuring better interoperability and accessibility. Further,

the application should be extended with other mining methods and libraries, e.g., high-utility mining or traditional itemset mining algorithms. Another important improvement would be to extend the data models to include more complex information and filtering options. The application should also be extended to support direct connections to databases instead of using the file system storage.

In conclusion, it could be shown that the data mining methods can help to discover and understand disease associations in large electronic health record datasets. Utilizing different pattern mining algorithms has the potential to change future research on disease comorbidities and improve clinical treatment.

BIBLIOGRAPHY

[1] D. Gligorijevic *et al.*, "Large-Scale Discovery of Disease-Disease and Disease-Gene Associations," (eng), *Scientific reports*, vol. 6, p. 32404, 2016.

[2] M. Flores, G. Glusman, K. Brogaard, N. D. Price, and L. Hood, "P4 medicine: how systems medicine will transform the healthcare sector and society," (eng), *Personalized medicine*, vol. 10, no. 6, pp. 565–576, 2013.

[3] C. C. Aggarwal, *Data mining: The textbook*. Cham, Heidelberg, New York, Dordrecht, London: Springer, 2015.

[4] P. Fournier-Viger, J. C.-W. Lin, R. U. Kiran, Y. S. Koh, and R. Thomas, "A Survey of Sequential Pattern Mining," 2017.

[5] R. Agrawal and R. Srikant, "Mining sequential patterns," in *Data Engineering, 11th International Conference (ICDE '95)*, Taipei, Taiwan, Feb. 1995, pp. 3–14.

[6] R. S. Rakesh Agrawal, "Fast algorithms for mining association rules," pp. 487–499, 1994.

[7] P. Fournier-Viger *et al.*, "A survey of itemset mining," *WIREs Data Mining Knowl Discov*, vol. 7, no. 4, e1207, 2017.

[8] C. Chand, A. Thakkar, and A. Ganatra, "Sequential Pattern Mining : Survey and Current Research Challenges," 2012.

[9] S. Abbasghorbani and R. Tavoli, "Survey on sequential pattern mining algorithms," in *Conference proceedings of 2015 2nd International Conference on Knowledge-Based Engineering and Innovation (KBEI): Tehran, Iran, Nov. 5-6th, 2015*, Tehran, Iran, 2015, pp. 1153–1164.

[10] M. J. Zaki, "Scalable algorithms for association mining," *IEEE Trans. Knowl. Data Eng.*, vol. 12, no. 3, pp. 372–390, 2000.

[11] J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation," *SIGMOD Rec.*, vol. 29, no. 2, pp. 1–12, 2000.

[12] J. Han, J. Pei, and X. Yan, "Sequential Pattern Mining by Pattern-Growth: Principles and Extensions*," in *Studies in Fuzziness and Soft Computing*, vol. 180, *Foundations and Advances in Data Mining*, W. Chu and T. Y. Lin, Eds., Berlin Heidelberg: Springer-Verlag GmbH, 2005, pp. 183–220.

[13] W. Gan, C.-W. Lin, P. Fournier-Viger, H.-C. Chao, and P. S. Yu, "A Survey of Parallel Sequential Pattern Mining," *undefined*, 2018.

[14] W. Gan, J. C.-W. Lin, P. Fournier-Viger, H.-C. Chao, and P. S. Yu, "A Survey of Parallel Sequential Pattern Mining," May. 2018. [Online] Available: http://arxiv.org/pdf/1805.10515v1.

[15] C. H. Mooney and J. F. Roddick, "Sequential pattern mining -- approaches and algorithms," *ACM Comput. Surv.*, vol. 45, no. 2, pp. 1–39, 2013.

[16] J. Pei, J. Han, and W. Wang, "Constraint-based sequential pattern mining: the pattern-growth methods," *J Intell Inf Syst*, vol. 28, no. 2, pp. 133–160, 2007.

[17] M. Garofalakis, R. Rastogi, and K. Shim, "Mining sequential patterns with regular expression constraints," *IEEE Trans. Knowl. Data Eng.*, vol. 14, no. 3, pp. 530–552, 2002.

[18] H.-E. Chueh, "Mining Target-Oriented Sequential Patterns with Time-Intervals," *undefined*, 2010.

[19] U. Yun and K. H. Ryu, “Approximate weighted frequent pattern mining with/without noisy environments,” *Knowledge-Based Systems*, vol. 24, no. 1, pp. 73–82, 2011.
[20] Helen Pinto, Helen Pinto, Simon Fraser University, “Multi-Dimensional Sequential Pattern Mining,” 2001.
[21] N. T. Nguyen *et al.*, *Eds.*, *Mining Frequent Itemsets from Multidimensional Databases: Intelligent Information and Database Systems*: Springer Berlin Heidelberg, 2011.
[22] Y. Liu, J. LI, W.-k. Liao, A. Choudhary, and Y. SHI, “HIGH UTILITY ITEMSETS MINING,” *Int. J. Info. Tech. Dec. Mak.*, vol. 09, no. 06, pp. 905–934, 2010.
[23] Y. Liu, W.-k. Liao, and A. Choudhary, “A Two-Phase Algorithm for Fast Discovery of High Utility Itemsets,” in *Lecture notes in computer science Lecture notes in artificial intelligence*, vol. 3518, *Advances in knowledge discovery and data mining: 9th Pacific-Asia conference, PAKDD 2005, Hanoi, Vietnam, May 18 - 20, 2005 ; proceedings*, T. B. Ho, Ed., Berlin: Springer, 2005, pp. 689–695.
[24] C. K.-S. Leung, “Uncertain Frequent Pattern Mining,” in *Frequent pattern mining*, C. C. Aggarwal and J. Han, Eds., Cham: Springer, 2014, pp. 339–367.
[25] D. C. Anastasiu, J. Iverson, S. Smith, and G. Karypis, “Big Data Frequent Pattern Mining,” in *Frequent pattern mining*, C. C. Aggarwal and J. Han, Eds., Cham: Springer, 2014, pp. 225–259.
[26] A. Grama, *Introduction to parallel computing*, 2nd ed. Harlow: Pearson, 2011.
[27] J. Dean and S. Ghemawat, “MapReduce,” *Commun. ACM*, vol. 53, no. 1, p. 72, 2010.
[28] S. G. Jeffrey Dean, *MapReduce: simplified data processing on large clusters*.
[29] P. Fournier-Viger *et al.*, “The SPMF Open-Source Data Mining Library Version 2,” in *Lecture notes in computer science Lecture notes in artificial intelligence*, vol. 9853, *Machine learning and knowledge discovery in databases: European conference, ECML PKDD 2016, Riva del Garda, Italy, September 19-23, 2016 : proceedings*, P. Frasconi, N. Landwehr, G. Manco, and J. Vreeken, Eds., Switzerland: Springer, 2016, pp. 36–40.
[30] J. Ayres, J. Flannick, J. Gehrke, and T. Yiu, “Sequential PAttern mining using a bitmap representation,” in *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, Edmonton, Alberta, Canada, 2002, p. 429.
[31] Xifeng Yan, Jiawei Han, Ramin Afshar, *CloSpan: Mining Closed Sequential Patterns in Large Datasets*.
[32] P. Tzvetkov, X. Yan, and J. Han, “TSP: mining top-K closed sequential patterns,” in *Third IEEE International Conference on Data Mining: Proceedings : ICDM 2003 : 19-22 November, 2003, Melbourne, Florida*, Melbourne, FL, USA, 2003, pp. 347–354.
[33] J. Wang and J. Han, *BIDE: Efficient Mining of Frequent Closed Sequences*: IEEE Computer Society.
[34] A. Gomariz, M. Campos, R. Marin, and B. Goethals, “ClaSP: An Efficient Algorithm for Mining Frequent Closed Sequences,” in *Lecture Notes in Computer Science / Lecture Notes in Artificial Intelligence*, v.7818, *Advances in Knowledge Discovery and Data Mining: 17th Pacific-Asia Conference, PAKDD 2013, Gold Coast, Australia, April 14-17, 2013, Proceedings, Part I*,

D. Hutchison, T. Kanade, and J. Kittler, Eds., Berlin/Heidelberg: Springer Berlin Heidelberg, 2013, pp. 50–61.
[35] P. Fournier-Viger, C.-W. Wu, and V. S. Tseng, “Mining Maximal Sequential Patterns without Candidate Maintenance,” in *Lecture Notes in Computer Science / Lecture Notes in Artificial Intelligence*, v.8346, *Advanced Data Mining and Applications: 9th International Conference, ADMA 2013, Hangzhou, China, December 14-16, 2013, Proceedings, Part I*, M. Yao, W. Wang, and O. Zaiane, Eds., Berlin/Heidelberg: Springer Berlin Heidelberg, 2013, pp. 169–180.
[36] I. Miliaraki, K. Berberich, R. Gemulla, and S. Zoupanos, “Mind the gap: large-scale frequent sequence mining,” *undefined*, 2013.
[37] P. Fournier-Viger, A. Gomariz, T. Gueniche, E. Mwamikazi, and R. Thomas, “TKS: Efficient Mining of Top-K Sequential Patterns,” in *Lecture Notes in Computer Science / Lecture Notes in Artificial Intelligence*, v.8346, *Advanced Data Mining and Applications: 9th International Conference, ADMA 2013, Hangzhou, China, December 14-16, 2013, Proceedings, Part I*, M. Yao, W. Wang, and O. Zaiane, Eds., Berlin/Heidelberg: Springer Berlin Heidelberg, 2013, pp. 109–120.
[38] V. S. Tseng *et al., Eds., Fast Vertical Mining of Sequential Patterns Using Co-occurrence Information: Advances in Knowledge Discovery and Data Mining*: Springer International Publishing, 2014.
[39] P. Fournier-Viger, C.-W. Wu, A. Gomariz, and V. S. Tseng, “VMSP: Efficient Vertical Mining of Maximal Sequential Patterns,” in *Lecture Notes in Computer Science / Lecture Notes in Artificial Intelligence*, v.8436, *Advances in Artificial Intelligence: 27th Canadian Conference on Artificial Intelligence, Canadian AI 2014, Montréal, QC, Canada, May 6-9, 2014. Proceedings*, D. Hutchison, T. Kanade, and J. Kittler, Eds., Cham: Springer International Publishing, 2014, pp. 83–94.
[40] F. Fumarola, P. F. Lanotte, M. Ceci, and D. Malerba, “CloFAST: closed sequential pattern mining using sparse and vertical id-lists,” *Knowl Inf Syst*, vol. 48, no. 2, pp. 429–463, 2016.
[41] K. Beedkar and R. Gemulla, “LASH: Large-Scale Sequence Mining with Hierarchies,” *undefined*, 2015.
[42] F. Petitjean, T. Li, N. Tatti, and G. I. Webb, “Skopus: Mining top-k sequential patterns under leverage,” *Data Mining and Knowledge Discovery*, vol. 30, no. 5, pp. 1086–1111, 2016.
[43] D. Tomar and S. Agarwal, “A survey on Data Mining approaches for Healthcare,” *IJBSBT*, vol. 5, no. 5, pp. 241–266, 2013.
[44] R. Bhardwaj, A. R. Nambiar, and D. Dutta, “A Study of Machine Learning in Healthcare,” in *2017 IEEE 41st Annual Computer Software and Applications Conference: 4-8 July 2017, Torino, Italy : proceedings*, Turin, 2017, pp. 236–241.
[45] *ICD-9-CM Diagnosis Codes - International Classification of Diseases - Medical Diagnosis Codes.* [Online] Available: https://www.findacode.com/icd-9/icd-9-cm-diagnosis-codes.html. Accessed on: Jan. 22 2019.
[46] W.-Q. Wei *et al.*, “Evaluating phecodes, clinical classification software, and ICD-9-CM codes for phenome-wide association studies in the electronic health record,” (eng), *PloS one*, vol. 12, no. 7, e0175508, 2017.

[47] *Java(TM) EE 7 Specification APIs.* [Online] Available: https://docs.oracle.com/javaee/7/api/overview-summary.html. Accessed on: Mar. 05 2019.
[48] *Bulma: Free, open source, & modern CSS framework based on Flexbox.* [Online] Available: https://bulma.io/. Accessed on: Mar. 05 2019.
[49] The jQuery Foundation, *jQuery.* [Online] Available: https://jquery.com/. Accessed on: Mar. 05 2019.
[50] Webflow, Inc., *Webflow.* [Online] Available: https://webflow.com. Accessed on: Mar. 06 2019.
[51] JetBrains, *IntelliJ IDEA.* [Online] Available: https://www.jetbrains.com/idea/. Accessed on: Mar. 10 2019.
[52] Software Freedom Conservancy, *Git.* [Online] Available: https://git-scm.com/. Accessed on: Mar. 10 2019.
[53] The Apache Software Foundation, *Maven.* [Online] Available: https://maven.apache.org/. Accessed on: Mar. 10 2019.
[54] Vitaliy Ostapchuk, *Disease Pattern Miner: vitaliy-ostapchuk93/disease-pattern-miner.* [Online] Available: https://github.com/vitaliy-ostapchuk93/disease-pattern-miner. Accessed on: Mar. 10 2019.
[55] The Apache Software Foundation, *Commons IO.* [Online] Available: https://commons.apache.org/proper/commons-io/. Accessed on: Mar. 10 2019.
[56] Google, *Guava: Projects - opensource.google.com.* [Online] Available: https://opensource.google.com/projects/guava. Accessed on: Mar. 10 2019.
[57] *StreamEx: amaembo/streamex.* [Online] Available: https://github.com/amaembo/streamex. Accessed on: Mar. 10 2019.
[58] *Tippy.js: Vanilla JS Tooltip and Popover Library.* [Online] Available: https://atomiks.github.io/tippyjs/. Accessed on: Mar. 10 2019.
[59] *Plotly.* [Online] Available: https://plot.ly/. Accessed on: May 01 2019.
[60] M. Bostock, *Data-Driven Documents - D3.js.* [Online] Available: https://d3js.org/. Accessed on: Mar. 10 2019.
[61] Mike Bostock, *Sankey Diagram.* [Online] Available: https://bost.ocks.org/mike/sankey/. Accessed on: Mar. 10 2019.
[62] Neil Atkinson, *Bi-directional hierarchical sankey diagram.* [Online] Available: http://bl.ocks.org/Neilos/584b9a5d44d5fe00f779. Accessed on: Mar. 10 2019.
[63] The Apache Software Foundation, *Apache Tomcat.* [Online] Available: http://tomcat.apache.org/. Accessed on: Mar. 13 2019.
[64] Continuous Delivery Foundation, *Jenkins.* [Online] Available: https://jenkins.io/. Accessed on: Mar. 13 2019.
[65] *Vue.js.* [Online] Available: https://vuejs.org/. Accessed on: Mar. 18 2019.
[66] *Express: Node.js web application framework.* [Online] Available: https://expressjs.com/. Accessed on: Mar. 18 2019.
[67] Docker Inc., *Docker: Enterprise Application Container Platform.* [Online] Available: https://www.docker.com/. Accessed on: Mar. 18 2019.
[68] P. Fournier-Viger *et al., PFPM: Discovering Periodic Frequent Patterns with Novel Periodicity Measures*: IntechOpen. Available: https://www.intechopen.com/citation-pdf-url/53394.

APPENDICES

Appendix A

Table X. Statistics for the dataset of interest.

AGE	GENDER	TRANSACTIONS	FILE SIZE	DISTINCT PATIENT IDS	DISTINCT ICD-CODES	VALID SEQUENCES	MAX SEQ-LENGTH	AVERAGE SEQ-LENGTH
0	MALE	330.472	10 MB	5.583	743	5.575	145	52,32
	FEMALE	274.547	8 MB	4.976	748	4.968	150	51,40
1	MALE	734.396	22 MB	22.608	930	22.527	172	38,23
	FEMALE	1.090.211	33 MB	30.114	1.008	30.058	179	43,77
2	MALE	1.333.104	41 MB	41.364	1.042	41.127	181	37,63
	FEMALE	3.265.206	98 MB	69.591	1.165	69.502	193	50,28
3	MALE	3.581.801	110 MB	83.313	1.140	82.972	216	48,61
	FEMALE	5.872.984	178 MB	107.517	1.231	107.390	215	55,82
4	MALE	8.110.348	254 MB	164.338	1.210	163.712	274	58,14
	FEMALE	11.510.832	355 MB	184.049	1.306	183.765	236	66,84
5	MALE	10.284.292	326 MB	173.837	1.243	173.306	271	67,96
	FEMALE	15.512.912	488 MB	206.373	1.316	206.141	242	80,42
6	MALE	14.520.204	462 MB	184.809	1.296	184.482	250	78,31
	FEMALE	19.637.099	626 MB	222.758	1.330	222.523	258	86,68
7	MALE	15.586.085	500 MB	162.695	1.286	162.432	257	82,96
	FEMALE	14.544.940	467 MB	151.183	1.300	150.947	261	86,99
8	MALE	3.184.040	102 MB	36.100	1.134	36.000	270	75,35
	FEMALE	3.287.923	106 MB	39.512	1.152	39.360	224	76,46
9	MALE	48.435	2 MB	866	587	852	186	54,68
	FEMALE	67.610	2 MB	1.298	621	1.276	194	52,71
0-9	MALE	57.713.177	1.828 GB	875.513	1.369	872.985	222,20	59,42
0-9	FEMALE	75.064.264	2.361 GB	1.017.371	1.396	1.015.930	215,20	65,14
0-9	ALL	132.777.441	4.189 GB	1.892.884	1.660	1.888.915	218,70	62,28